Chapter Five: Contents –

(Mode Choice – 10 December 2002 – LA-UR 01-5713 – Portland Study Reports)

Chapter Five—Mode Calibration and Assignment in the Portland Study

1. INTRODUCTION

Mode calibration and assignment in the Portland Study is a demonstration of a general methodology. It is limited to only those travelers on work tours that have home and work locations near transit stops. The procedures shown in this document demonstrates an implementation of the general mode methodology (**GMD**) that is given in TRANSIMS Ver. 3, Volume Seven (*Methods in TRANSIMS*), Chapter One (*Mode Choice*). It is advised that the general documentation be read before attempting to understand the results outlined in this document.

A non-logit mode choice methodology is used in the Study. It is a simple methodology based on fitted semi-parametric logit like functions of costs that are calibrated to the survey mode splits. Mathematically, it follows the results shown in Section 2.1 (*Mode Choice Theory*) of the **GMD**, and in that sense is a valid methodology. It should be noted that if a logit is thought of as a "behavioral" model, then the models here are also "behavioral" models as they too are composed of statistical fits to human mode choice "behavior" data.

There are numerous research reasons for the development of this methodology for the Study rather than using a logit approach. These reasons include:

1) *To study a limited fidelity mode choice methodology.* The obvious methodology is to fit a logit. A second methodology is a series of logit-like fits to subpopulations of travelers. These types of models require less modeling assumptions and, as such, are better suited for forecasts. One of the purposes in the Study was to ascertain whether a collection of these simplified models can be calibrated.

2) *To understand TRANSIMS*. Rather than spend time making statistical fits to survey data, it was deemed more important to give guidance on the behavior of the Route Planner when making mode choices. Also, the general implications of assigning modes to individuals for an entire day and simulating their movements on a second-by-second basis needed to be understood. The general theory of mode calibration given in TRANSIMS Ver. 3, Volume Seven (*Methods in TRANSIMS*), Chapter One (*Mode Choice*), Section 2.1 (*Mode Choice Theory*) is a result of these studies.

3) *To demonstrate that many methodologies can be implemented and used in the TRANSIMS Framework*. Other than the Route Planner behavior, all of the results of Section 2.1 of the **GMD** are dependent on user-defined mode assignment functions. TRANSIMS, being a modeling system rather than a model, is shown to support many methodologies for mode choice.

4) *To understand mode assignment in tours*. Consistency of mode choice in tours and the calibration of modes by tours needed to be understood.

5) *To exercise the TRANSIMS Collator*. The TRANSIMS technology that brings together the activity list, the routes, and the microsimulation results is the Collator. The mode choice experiment allowed testing and revising of the Collator so that information needed to make mode choice is available to the user regardless of the methodology used.

The mathematical notation and the procedures for mode calibration used in this document follow that of the **GMD**. The methodology in that document calls for the determination and calibration of the functional form for the choice functions that represent the probabilities $P_U^{(1)}(M_1|D)$, $P_U^w(M_2 = A|s_w, D)$, $P_U^b(M_2 = A|s_b, D)$, and $P_U^l(M_2 = A|s_l, D)$.

Here $P_U^{(1)}(M_1|D)$ is a first stage choice procedure that chooses between the modes unchanged by the "rational choice" of the Route Planner, bike (*BI),* school bus (*S),* inter-household shared rides (*M*), and those that may change denote by the combined mode *X.* The mode *X* represents the vehicular and walk modes which, because of the "rational choice" behavior of the Route Planner, needs separate calibration. The mode *X* could also include park and ride, but it is probably better to calibrate it as a separate mode in stage 1. The stage 2 choice procedures represent the probabilities choosing auto or transit depending on the behavior of the Route Planner when it routes the trip by generalized transit (*T*). In the **GMD** these probabilities are $P_U^w(M_2 = A|s_w, D)$, $P_U^b(M_2 = A|s_b, D)$, and $P_U^l(M_2 = A|s_l, D)$, and represent probabilities for three mutually exclusive mode splits—walk legs only, at least one bus leg, and only light rail legs—of the trips/tours for these routes. As pointed out in the **GMD**, any split of these trips/tours is allowed as long as the splits are mutually exclusive.

In this demonstration, no attempt is made to calibrate the first stage of the mode choice, $P_U^{(1)}(M_1|D)$. Two mutually exclusive partitions of the trips/tours are used for the second stage calibration. The first, s_w, is a collection of those trips/tours that contain only walk legs when routed as general transit (*T*). The set s_t contains those trips/tours with at least one transit leg. Simple functional forms are used for the choice functions representing the probabilities $P_U^w(M_2 = A|s_w, D)$ and $P_U^t(M_2 = A|s_t, D)$. It is the calibration of the functions representing the probabilities $P_U^w(M_2 = A|s_w, D)$ and $P_U^t(M_2 = A|s_t, D)$ that is of interest here, as these functions are influenced by the "rational choice" behavior of the Route Planner.

2. CASE STUDY MODE CHOICE

A mode choice methodology is developed in the **GMD**. In the Study, a limited fidelity model with few assumptions is used to calibrate a mode choice for a subset of the population. The following subsections detail the user-defined mode assignment functions that are used in the context of the general methodology. Enough detail is given so that the reader may reproduce the results and fine-tune the methodology to make it viable for the general transportation community as well as for research purposes.

2.1 Case Study Model & Definitions

In the Study, no calibrated choice functions are generated to represent the probability $P_U^{(1)}(M_1 \mid D)$ functions in choosing among modes *BI, S, M*, or *X*. For this demonstration, it is assumed that the mode choice assigned by the TRANSIMS Activity Generator for the bike, school bus, and shared-ride trips is reasonable. In practice, $P_U^{(1)}(M_1 \mid D)$ would either be calibrated or, at a minimum, the mode choice for *BI, S, M*, or *X* assigned by the Activity Generator would be checked. In the Study, when a trip using bike or school bus mode is included in an otherwise transit, walk, or auto tour, only the transit, walk, and auto trips in the tour are changed—the bike and school bus trips are unchanged. Given the complexity of intra-household shared rides, tours involving them are not considered in the mode choice procedures in the Study.

All mode assignments in the Study are estimated for home-to-home tours, not from trips. Despite the fact that trips are a very convenient atomic unit for measurement and for mathematics, in reality no traveler considers a trip independent of the rest of the tour. Since TRANSIMS is a disaggregate representation of individual traveler's activities, complete tours are its basic feature. Also, mode choice makes sense only on a tour basis since mode choice involving only trips would require changes to the activity pattern of the individual traveler. For example, without access to a non-personal auto, a traveler on a bus tour could not have an auto trip placed in the middle of the tour without walking all the way home to get the car—which would add a trip to the tour.

It is not unusual in current transportation practice to develop mode calibration functions, usually logits, for large subsets of the tours. For example separate logit calibrations are developed for home-based work tours, home-based other tours, etc. The method of modeling in the Study is to use smaller groups of tours and simple, calibrated one-parameter models for determining mode choice. A different value for the parameter is estimated for each group of traveler tours. Tours are placed in groups of similar tours, defined as tours that are likely to produce nearly the same calibration parameter. Calibration is done independently for each group. The Study demonstrates this procedure for one small group of like tours.

The procedures for the Study follow the steps outlined in the **GMD**. The calibration stage is completed in the Study by sampling tours from the base year activity set. It could be just as well accomplished by locating activities from the activity survey on the base year

TRANSIMS network and calibrating each grouping from the trips between the activity locations. This is the usual approach in transportation mode choice modeling. In the TRANSIMS Framework, the survey activities are located on the TRANSIMS network so that the Route Planner's "rational choice" behavior can be estimated.

The calibration and application steps used in the Study are:

1) Groupings of tours are determined that are likely to have, on a statistical basis, the same mode choice characteristics. Additionally, the form of a calibration function is determined. The groupings and the calibration function are user-defined choices and are based on the modeler's knowledge and past determinations of those factors that influence mode choices.

2) A calibration strategy for the first stage, $P_U^{(1)}(M_1 \mid D)$, is determined. In the Study, the mode assignments from the Activity Generator for the modes *BI, S, M*, and *X* are taken unchanged.

3) The modeler decides on the mutually exclusive partitions of the tour routes when the *X* mode tours are routed using the general transit mode *T*. Two partitions are used in the Study. The first, s_w, is for tours that contain only walk legs. The second, s_t, is those tours that contain at least one transit (either bus or light rail) leg.

The following second stage calibration and application steps are completed for each of the tour groupings. Steps 4-7 are the calibration stage.

4) A random sample of the tours, all of type *X*, for each grouping is drawn from the base year activity set.

5) Each of the selected tours is routed with both the auto mode (*A*) and the general transit mode (*T*). These routings produce travel characteristics such as travel times and monetary costs for the two modes. Additionally, routing by *T* partitions the sample into mutually exclusive sets depending on the "rational choice" behavior of the Route Planner. In the Study, two sets are considered—those tours will only walk legs, s_w, and those with at least one transit leg, s_t. The proportions, p_R^w and p_R^t (see Section 2.2.1 of the **GMD**), are estimated from the proportion of tours that are classified as s_w or s_t.

6) The target values for auto, walk, and transit are determined from the calibration data and the estimates of p_R^w and p_R^t. (See equations 14 to 16 of the **GMD**.)

7) Two sets of calibration parameters (in the Study each set consists of only one parameter) are determined. The first allows for choice between auto (*A*) and generalized transit (*T*) given that the Route Planner "chooses" the partition s_w; that is, that the tours contain only walk legs. The second allows for choice between auto (*A*) and generalized transit (*T*) given that the Route Planner "chooses" the partition s_t. In this case, the choice is between transit legs and auto. It should be noted that these data could be used to calibrate a logit choice model for both partitions. Fitting a logit here

implies that the modeler believes that the probability of auto or transit (*A* or *T*) is a monotone function of some linear combination of the variables not used to stratify the groupings. In the Study, we use a semi-parametric approach similar to that shown in the **GMD**.

Steps 8-10 are for applying the calibrated second stage choice functions to a forecast year.

8) A random sample of tours of the specified type or grouping is selected from the forecast year activity set. These tours are routed both in the *A* and *T* modes. The costs associated with these tours are retained, and mode assignment for the forecast year is based on them. If complete parametric calibration fits, such as logits, are produced in step 7, then this step is not necessary.

9) Using the above parametric or nonparametric calibrations, mode choices are made for each tour in the forecast population.

10) All of the above procedures are statistical in nature, and the quality of the calibration is judged on a population basis. In individual cases, the mode assignment may be unrealistic. If there are a great number of these cases, they need to be iterated to produce more realistic results. For example, an analyst may wish to change the modes of travelers with more than three or four transfers on transit trips to auto. In this process, a similar number of travelers whose mode is auto would have a mode change to transit. Care must be taken in these iterations to maintain calibration.

The variables considered for stratification and the choice function used in the Study are described in the next sections. In the Study, one strata is chosen for analysis. A random sample of 99 tours from that strata for the base year is selected for calibration. Mode choice is calibrated for this strata using a simplified cost function. A second sample of 100 tours is selected from the base year activity set as a surrogate for the forecast year. Modes are assigned to this sample using the calibration determined from the first sample.

2.1.1 Model: Cost Function and Stratification

The mode choice modeling in the Study is in two stages—development of a generalized cost function and a set of stratification variables. The basic approach assumes that if a tour's generalized cost on each of two modes is evaluated, then the tour will be more likely to "prefer" the one with the lower cost. The cost function itself, however, is only one stage of the modeling. The other stage is stratifying the data to create subgroups of tours that are similar in their mode choice characteristics. The choice of the stratification parameters implicitly includes them in the mode utility without being included explicitly in the generalized cost function.

In preparing the stratification and cost function for the Study, all of the variables listed in Table 1 were considered.

Table 1. Potential mode-choice parameters considered.

Variables
work/non-work tours
travel time
distance traveled
household income
river crossings
CBD/non-CBD
distance to transit
dollar costs
urbanization of locations
age
departure time
time of day
tour complexity
fraction of workers in HH
mode reliability
mode "comfort"
presence of children
single/multiple-family dwelling

Clearly, some of these variables are redundant, and some are unsuitable for modeling in a forecast setting. It was decided that the travel time, dollar-cost, and household income should be included in the cost function, in part because they were deemed important for the mode decision, but also because they lend themselves most easily to the continuous nature of the cost function without further parameterization. The tour type (work, non-work), home urbanization (very urban, urban, or suburban), and primary anchor urbanization were selected to be part of the stratification. The "effective" utility for mode choice, therefore, includes time, cost, income, home and anchor urbanizations, and tour type.

2.1.1.1 Cost Function

The generalized cost function used in the Study is

$$c_i = \alpha \cdot \log\left(1.01 + \min\left(0, Income\right)\right) \cdot Time_i + DollarCost_i$$

for mode *i*, where the $Time_i$ is the total time spent traveling between locations on a tour, including all trips. *Income* is the household income, although any negative number is made zero and 1.01 is added to keep the logarithm well behaved and the mode choice consistent with respect to time. The $DollarCost_i$ is a measure of the monetary cost of the tour by mode *i*. This function exhibits the trade-off between the monetary cost for traveling by a mode and the time it takes to make the trip using the mode. The relationship between a traveler's time and monetary cost is weighted by both the household income and the calibration parameter α. The same cost function is used to compare the utility of different modes by using their respective time and dollar-costs in the function.

2.1.1.2 Stratification

Mode calibration and application in the Study is demonstrated using one strata. This strata is comprised of those tours where the primary tour purpose is work. Additionally, the home and work locations are in very urbanized areas. The total number of strata for the entire population is 18. This is the result of having two choices for tour type and three each for the home and primary anchor urbanizations. Each of the 18 strata would have its own calibrated values of α. These selected stratification makes use of naturally discrete variables, so no additional parameters are required to further split the stratifications.

Using the assumption that tours whose activity locations are far from transit will be highly unlikely to use transit, another stratification is introduced to determine mode-choice rule-sets. Tours are stratified by the distance-to-transit of the home and primary locations on the tour. The assumption is that no one will walk for several hours to take a bus, but that if the home location is far from transit but the primary anchor is not, they may use Park & Ride. The additional four strata are: origin near and destination near transit (ONDN), origin far from transit but destination near (OFDN), and the remainder of tours that all have destination far from transit (DF). For ONDN tours, travelers may choose the transit modes or auto mode; for OFDN tours, pure transit tours are excluded, but Park & Ride allowed; and for DF tours, transit is excluded altogether. Using these definitions, there are $2 \cdot 3 \cdot 3 \cdot 3 = 54$ strata, but the number simplifies. For all of the DF tours, the additional assumption may be made that walking, biking, or any transit mode is infeasible, leaving only auto modes for all tours. Under this assumption these tours do not need to be examined any further—all are assigned to auto tours.

Table 2. Calibration target mode splits determined from the survey.

work/non-work	home urbanization	work urbanization	transit distance	Walk	Bike	Transit	Park & Ride	Drive Alone	Total
work	very-urban	very-urban	ONDN	59	2	41	0	51	153
work	very-urban	very-urban	OFDN	0	0	0	0	0	0
work	very-urban	very-urban	DF	0	0	0	0	0	0
work	very-urban	urban	ONDN	14	3	2	0	39	58
work	very-urban	urban	OFDN	0	0	0	0	0	0
work	very-urban	urban	DF	0	0	0	0	0	0
work	very-urban	suburban	ONDN	0	2	9	1	52	64
work	very-urban	suburban	OFDN	0	0	0	0	0	0
work	very-urban	suburban	DF	0	0	0	0	0	0
work	urban	very-urban	ONDN	44	24	101	6	242	417
work	urban	very-urban	OFDN	0	0	0	0	0	0
work	urban	very-urban	DF	0	0	0	0	0	0
work	urban	urban	ONDN	23	13	31	0	256	323
work	urban	urban	OFDN	0	0	0	0	0	0
work	urban	urban	DF	0	0	0	0	0	0
work	urban	suburban	ONDN	7	2	21	0	430	460
work	urban	suburban	OFDN	0	0	0	0	0	0
work	urban	suburban	DF	1	2	0	0	41	44
work	suburban	very-urban	ONDN	33	8	60	140	414	655
work	suburban	very-urban	OFDN	1	0	2	25	54	82
work	suburban	very-urban	DF	0	0	0	0	0	0
work	suburban	urban	ONDN	10	4	12	7	372	405
work	suburban	urban	OFDN	1	0	0	0	59	60
work	suburban	urban	DF	0	0	0	0	0	0
work	suburban	suburban	ONDN	65	19	43	12	2225	2364
work	suburban	suburban	OFDN	4	0	0	1	552	557
work	suburban	suburban	DF	17	3	0	0	380	400

work/non-work	home urbanization	work urbanization	transit distance	Walk	Bike	Transit	Park & Ride	Drive Alone	Total
non-work	very-urban	very-urban	ONDN	109	4	12	1	29	155
non-work	very-urban	very-urban	OFDN	0	0	0	0	0	0
non-work	very-urban	very-urban	DF	0	0	0	0	0	0
non-work	very-urban	urban	ONDN	13	5	7	0	26	51
non-work	very-urban	urban	OFDN	0	0	0	0	0	0
non-work	very-urban	urban	DF	0	0	0	0	0	0
non-work	very-urban	suburban	ONDN	4	0	8	0	13	25
non-work	very-urban	suburban	OFDN	0	0	0	0	0	0
non-work	very-urban	suburban	DF	0	0	1	0	1	2
non-work	urban	very-urban	ONDN	29	5	26	1	94	155
non-work	urban	very-urban	OFDN	0	0	0	0	0	0
non-work	urban	very-urban	DF	0	0	0	0	0	0
non-work	urban	urban	ONDN	138	16	21	0	295	470
non-work	urban	urban	OFDN	0	0	0	0	0	0
non-work	urban	urban	DF	0	0	0	0	0	0
non-work	urban	suburban	ONDN	16	2	9	0	170	197
non-work	urban	suburban	OFDN	0	0	0	0	0	0
non-work	urban	suburban	DF	1	0	0	0	11	12
non-work	suburban	very-urban	ONDN	11	0	12	8	73	104
non-work	suburban	very-urban	OFDN	1	0	0	1	17	19
non-work	suburban	very-urban	DF	0	0	0	0	0	0
non-work	suburban	urban	ONDN	14	1	4	0	200	219
non-work	suburban	urban	OFDN	0	0	0	0	14	14
non-work	suburban	urban	DF	0	0	0	0	0	0
non-work	suburban	suburban	ONDN	212	14	38	4	1527	1795
non-work	suburban	suburban	OFDN	1	0	0	0	244	245
non-work	suburban	suburban	DF	31	4	1	0	239	275
			Total	**828**	**129**	**460**	**207**	**7881**	**9780**

2.1.2 Calibration of this Model

In the Study, mode calibration is made using the non-logit form of the cost function shown in the previous section. Two mutually exclusive partitions of the "*T*" mode tours are used. These are tours with all walk legs and those with at least one transit leg.They are denoted by s_w and s_t. Therefore, each stratum has two parameters to be calibrated. There is a separate value of α in the cost function of Section 2.1.1.1 for each of the sets s_w and s_t. Each set of two parameters is for a distinct subset of tours in the stratum. The parameters are estimated with two one-dimensional fits—not a single two-dimensional fit. These fits are easily accomplished using any one-dimensional search algorithm. This is especially true because our experience has shown the relationship between the transit fraction and the calibration parameter α to be monotonic in most cases.

Two conditions can make calibration using the type of cost functions of Section 2.1.1.1 impossible. Consider two costs from the same tour routed by different modes. The costs determine which tours prefer the first mode and which prefer the second based on the mode that exhibits the smaller cost. A representation of this with the line showing where the costs are equal is shown in the first panel of Fig. 1. Here, the y-axis is the difference in travel times, $T_1 - T_2$; and the x-axis is a scaled difference in the costs for the tours, $(D_2 - D_1) / f(I)$. The points in this figure are plotted with a fixed value of the calibration parameter, α.When α is changed, the costs of the tours change and the points move. Some of the points can move above and some below the line simultaneously, so this is not a good way to visualize the effect of changing α. Instead, the cost function is broken into its component parts (time and monetary costs) and their differences examined

independent of α. This is shown in the second panel of Fig. 1. In this case, it is the line that splits the tours into those preferring mode-1 to mode-2 that moves, and the points themselves that are fixed. This plot has an additional benefit in that it clearly shows which tours are fixed to a single mode. Because α is non-negative, the line that splits the two groups must lie in the upper-right and lower-left quadrants of the plot. Any points in the upper-left and lower-right quadrants will not cross the line no matter the value of α. For example, in the upper-left quadrant of the figure, the time of the first mode is greater and the monetary cost is more, meaning that regardless of the value of α, mode-1 always has a lower utility according to the cost function. If too many points lie in these quadrants, the cost function cannot necessarily be calibrated to match a desired mode split.

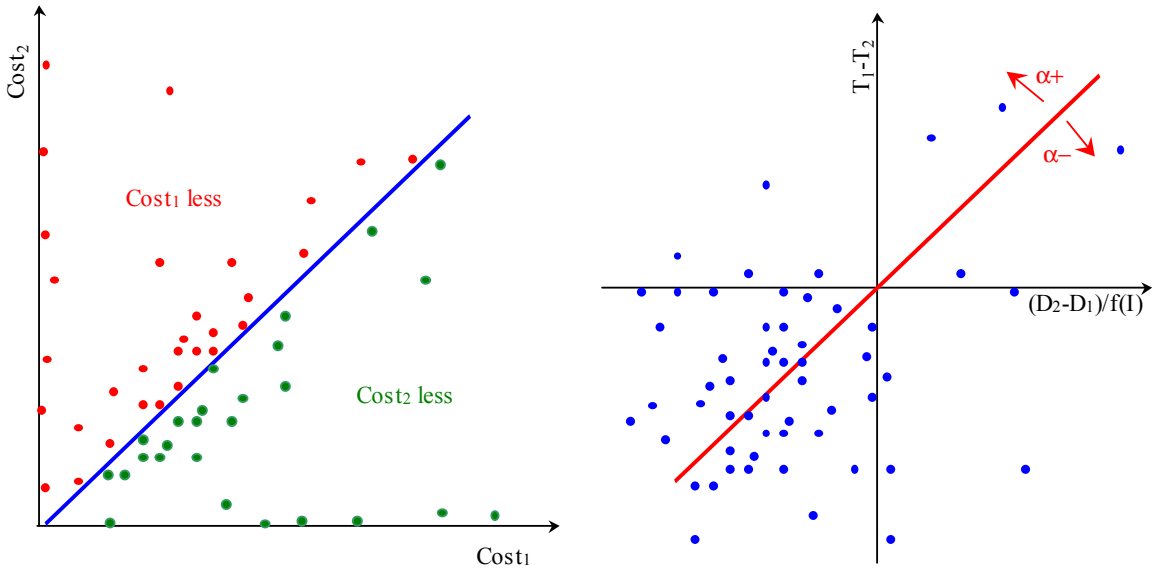

Fig. 1. Two ways to visualize the cost function comparisons.

A second condition that precludes calibration is if the angular distribution of tours is symmetric across the origin in the upper-right and lower-left quadrants of the second panel of Fig. 1. In this case, there will be a constant mode split independent of α. Since the trade-off is usually between time and monetary costs, however, this condition would rarely occur. Almost all strata will have tour costs concentrated in either the upper-right or lower-left—not both.

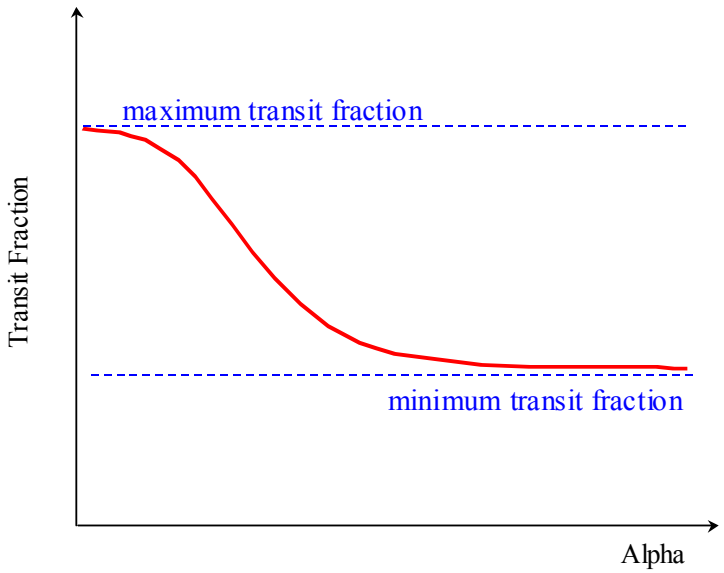

Fig. 2. Minimum and maximum mode splits available and the calibration curve as a function of α*. Calibration to transit target values outside the maximum and minimum transit fraction bounds is not possible..*

Both of the above conditions bound the possible fraction of transit tours. Fig. 2 illustrates the effect of both of these conditions. In this figure, the maximum and minimum possible transit target fractions and the calibration function as it changes with α are given. The two conditions cited above can tend to move the maximum and minimum target values together. In most cases, they will reduce the maximum and increase the minimum possible transit fraction, but the target will still lie safely in between. Only when the target fraction lies outside the possible range is the model unable to be calibrated for the particular stratum. When that happens, the user can do several things. In such cases, the stratum may be merged with a similar one, or if it is deemed that the model is incapable of describing features of that stratum, a new model or cost function could be developed particular to that stratum.

The proportion of tours where the cost is lower, using one of the generalized transit modes (walk only or at least one transit leg), than it is to drive is a function of the calibration constant α. There is no mathematical reason for this proportion to be a decreasing or increasing function of the calibration constant. This is apparent from the right-hand panel of Fig. 1. Since there are points in both the upper-right and the lower-left quadrants, new points appear above the line while others move below the line as α sweeps from left to right. In this case, the proportion of points above the line is not monotone with α. If, for a particular cost function, this lack of monotonicity is severe, then a new cost function must be devised since such cost functions are unsuitable for calibration. In those cases where there are no points in either the lower-left or the upper-right quadrants (but not both), then the calibration function is monotone with α.

The Study mode methodology considers home-to-home tours. Such a tour is defined as the sequence of trips and activities starting at the home location and returning to the home location. A second departure from the home location produces a second tour, which

is considered independently from the first (aside from demographic and home information being identical).

Tour calibration information is gathered into a tour-based database. A general tour-based database consists of four types of data: traveler/household demographics, whole-tour information, home information, and primary anchor information.

The scripts to generate this generalized database are found in Appendix B of this Chapter.

All of the fields are listed by-category in Table 3. However, not all of these variables are used to determine mode assignment in the Study.

Table 3. Tour database fields.

Demographic	Tour	Home	Primary Activity
HH	Tour Number	Zone	Activity ID
Traveler	Tour Act	River	Zone
HH Income	Tour Type	Urbanization	River Zone
Age	Tour Mode	Transit Distance	Urbanization
	Mode Preference		Transit Distance
	Time		Parking Zone
	Distance		Activity Mode
	MultiMode		
	Shared		
	SubTours		
	CrossColumbia		
	CrossWillamette		
	RailOnly		
	MaxTranDist		
	ModeString		

The demographic data is self-explanatory. Both the home and anchor locations have four zone classifications:

- the 1260-zone (**TAZ**) number;
- a river zone (where 1=North of the Columbia, 2= South of the Columbia, West of the Willamette, and 3= South of the Columbia, East of the Willamette);
- an urbanization value (1=very urban, 2=urban, and 3=suburban); and
- a Euclidean distance to the nearest transit stop.

The primary anchor has three other fields:

- the activity ID of the primary anchor,
- one of the seven parking zones defined below, and
- the mode of travel to the primary anchor activity.

The tour number is the unique identifier (with Traveler ID) for a tour. The tour activity is the first non-home activity on the tour, which is the required field for the Activity Regenerator to change all locations on a tour. For changing modes in the Activity Regenerator, any activity on the tour (such as the anchor activity) can be used.

The primary activity is classified as follows:

- If there are any work, school, or college type activities in the tour, then the longest among them is the primary activity, and the tour type is classified as "work". In the Study, school and college are excluded from calibration.
- If there are no work, school, or college activities, but there is at least one shopping activity, then the longest shopping activity is the primary activity, and the tour is a "shop" type tour.
- Otherwise, the longest duration activity is the primary activity, and the tour type is "other".

In the Study, the *ModePreference* for the tour is set as *mode* to the primary activity. When tours are classified according to destination distance to transit (ONDN, OFDN, DF), it is the primary activity that is considered to be the destination.

The tour travel time and distance are accumulated throughout the tour. For travel between each location, the values of the Route Planner's estimate of travel time and the Euclidean distance between locations are accumulated. The number of subtours in a tour is also accumulated. Each time a traveler returns to an activity location previously visited in the same tour, all activities in between are part of a subtour. Note that, in the Study, all information from subtours except the existence of shared auto rides is ignored.

The multi-mode field is non-zero if the tour makes use of more than one mode. The number of rail-only activity modes is also reported, as are the number of trips in the tour that crosses each of the rivers. The maximum distance to transit from among all activity locations is not currently used in mode feedback, but may be useful in mode feedback in the future.

The "shared" field of the tour database reports intra-household shared rides in a traveler's tour. The hierarchy is:

- 0 = individual trips only,
- 1 = passenger on subtour only,
- 2 = passenger on main tour,
- 3 = driver on subtour only, and
- 4 = driver on main tour.

Any occurrence of a trip with a higher number redefines the "shared" tour number. That is, if shared = 3, there are no main tour driver trips, but the traveler may or may not have been a passenger on the main tour.

The Collator collects the mode strings produced by the Route Planner for each trip. A typical example is that an auto trip may be wcw, where the w's are the walks to and from the auto, and the c is the driving part of the trip. When creating the tour-based database, all of the mode strings for non-subtour trips on the tour are concatenated to produce the tour mode string.

Table 4. Tour modes in the tour database.

Mode =	Description
1	walk, bike, or inter-household shared ride
2	auto
3	bus
4	rail
5	mixed transit
6	mixed auto and transit

Table 4 describes the numeric values assigned to the tour modes in the database. If the mode is bike only, school bus only, or inter-household only, then that mode exclusively defines the tour. Otherwise, the tour contains at least one trip of what has been called mode *X*. The tour mode string is used to determine the tour-mode. Any occurrences of inter-household shared rides, school bus, or bicycle mode are ignored. If there are all walks in the tour, the tour is a "walk" mode tour. If there is only walk with one of auto, bus, or rail, the non-walk mode defines the tour mode. (For example, a tour with modes auto-walk-walk-auto is classified as an auto tour.) If there is only walk, bus, and rail, the tour is "mixed transit". And, if there is any combination of walk, auto, and bus and/or rail, the tour is mixed auto and transit. For calibration and application purposes in the Study, mixed-transit is considered bus, and mixed-auto-and-transit is considered auto.

2.1.3 Travel Time Definition

The travel time used in the cost function is the in-motion time for an entire tour. This excludes time spent at an activity location, but includes all time spent in going between activity locations. For a home-work-home auto tour, it would include the time spent in the auto, as well as time spent walking to and from the auto.

Some trips are not routable, and the Collator reports "NA" for their travel times. An example of such a trip is an attempted walk when no walk path exists. If any trip on a tour has an NA travel time, then NA is reported for the entire tour. In the processing of that time for fitting or assigning modes, however, the NA is converted to 24 hours for calculation purposes.

Note that travel on subtours is not included in the accumulation of tour travel times.

2.1.4 Monetary Cost Details

The generalized cost function makes use of two types of monetary cost incurred on a tour. In the Study, all transit tours are assessed a cost of $1.00. It was found that all strata could be calibrated regardless of the value of the cost, as long as it was the same order of magnitude as $1.00. This does not mean that the mode split is insensitive to transit cost,

but that the calibration process is indifferent to it. With different transit costs, the cost function parameters will change, but a fit is still possible. The second cost in the Study is the auto cost. The average long- and short-term parking prices by Traffic Analysis Zone (TAZ)is used and is shown in Table 5. The fields in the table are the TAZ, the short- and long-term parking costs in dollars, and a description of the geographic location in the Portland area. In addition to parking costs, each driver is assessed $.03 for each kilometer traveled by auto.

It should be noted that the term $DollarCost_i$ in the generalized cost function used in the Study drives the choice of modes. This term reflects the balance between transit cost and driving costs. It is evident from Table 5 that very few zones have nonzero parking costs, so almost all of the driving costs are comprised of the $.03 per kilometer charge.

Table 5. Parking costs by location.

TAZ	Short	Long	Location
1,2,10-16	2.47	4.94	CBD South of Burnside
3-6	1.59	3.18	CBD North of Burnside
43	1.38	2.76	Oregon Health Sciences University
510,934-936	0.80	1.59	Oregon City
846-847	0.00	3.03	Lloyd District
971-981	0.85	1.70	Vancouver WA
	0.00	0.00	all other zones

Note that the costs described are for an entire tour. A particular traveler might use transit several times, or have an activity set that makes many auto trips in the central business district. In either case, the appropriate costs are incurred only once in the Study model.

2.1.5 Urbanization Value

The "Urbanization" indicator for an activity location is the sum of two values: retail employment within 1 mile, and the total number of employees whose homes are within 30 minutes transit travel time (including in-vehicle, wait, and walk time). In the Portland network used for the Study, the largest number of retail locations within a mile of transit is 17,709, and the largest number of possible employees within 30 minutes transit-distance is 267,874. The boundaries between subgroups are chosen to be as shown in Table 6. The three categories of urbanization are listed in the first column. In the second column are the ranges of values of urbanization defining that category, and in the last column contains the number of zones (TAZ) in each category.

Table 6. Urbanization values.

	Range of Values	Resulting Number of Zones
Very Urban	203168 – 283864	28
Urban	62649 – 203167	106
Suburban	0 – 62648	1113

2.1.6 "Near Transit"

A different set of mode choice rules is applied to people based on their distance to transit. In the Study, an activity location within 1000 meters (about a 15-minute walk) of a transit stop is considered to be "near transit".

The origin of the tour is defined as the home location, and the destination as the primary activity location (described in tour classification rules above). Hence, an ONDN tour is one with the home location within 1000 meters Euclidean distance of any transit stop and the primary activity location within 1000 meters Euclidean distance of any transit stop. This is no guarantee that every activity on the tour is near a transit stop, or that a trip between home and the primary destination are even connected through the transit system.

2.1.7 Other Notes

In all mode assignments in this Study, it is assumed that the non-auto, non-generalized transit modes are correct as chosen by the Activity Generator. In practice, these mode assignments would be made through a calibrated first stage choice function.

In the calibration of the ONDN strata, some additional data filtering is performed. All tours involving any shared ride component are ignored. Even the inclusion of tours having only a passenger shared-ride trip on a subtour would have required carefully reinstating the subtour modes. Repairing damage to other types of shared rides would require substantially more modeling. In addition, only travelers over the age of 15 are included, since driving is not an option for those under that age—they must either be on (generalized) transit or be a shared-ride passenger. If one of these travelers has an infeasible transit tour, then the activity locations must be changed.

Some transit trips, including walks, are considered to be unfeasible due to their length. Unfeasible transit is defined in the Study as any transit trip over 24 hours or having more than three transfers.

Any tour that is a one-stop walking tour with a very short duration at a social/recreational activity type is left as it is in the survey. These represent recreational walks or jogs and should not have their modes changed; however, some may require an activity location change.

2.2 Results of Calibration and Application

The mode choice methodology is illustrated in this section for one of the strata, very urban to very urban ONDN work tours. The calibration target values for this particular strata are given in the first line in Table 2. This results section is split into the calibration stage and those related to application. Both stages are described in the preceding sections of this document and the GMD. The non-logit cost function described in Section 2.1.1.1 is used as the basis for mode choice.

In the calibration stage, 99 tours are selected at random from the very urban to very urban ONDN work tours. Only tours with the "*X"* type mode to the primary work anchor are

considered. To facilitate routing, a new activity set is created for the tours in the sample. In this activity set, the selected tours comprise the entire activity set for the traveler. Modes of subtours of the main tour are all set to inter-household shared ride, or mode=8 in the Study.

The application stage is demonstrated using a sample of 100 tours. Three methodologies for mode choice using the non-logit form of the calibration function are demonstrated. It is shown that calibration is maintained for the application tours when routing is completed by: both auto (*A*) and generalized transit (*T*), generalized transit alone, or auto alone.

2.2.1 Initialization and Calibration

The calibration sample of 99 tours is selected from the ONDN/work/home in very-urban/work in very-urban stratum. This strata is denoted as ONDN w.1.1. These tours satisfy all of the conditions listed previously; that is, non-drivers and non-simulated modes (e.g., bikes, school buses, and inter-household shared rides) are not selected. There are 2556862 tours in the activity set: 1945475 are within 1 km of transit for both the origin and destination and are in the ONDN classification. Of those, 7634 tours are in the work (or college/school), ONDN w.1.1, stratum. On eliminating those tours with shared rides, travelers under the age of 16 and college/school as the primary anchor, 5407 tours remain. The calibration sample of 99 tours is selected from these.

2.2.1.1 Cost Function and Target Values

The cost function for the calibration is given by:

$$c_i = \alpha \cdot \log\left(1.01 + \min\left(0, Income\right)\right) \cdot Time_i + DollarCost_i$$

as described in Section 2.1.1.1. Here, $Time_i$ is the travel time for the primary tour, excluding any subtour times. The $DollarCost_i$ is \$1.00 for all transit tours, and for auto trips it is \$.03 per km plus any parking cost. There is no $DollarCost_i$ for walking trips. *Income* is the household income, and α is the calibration parameter. The values for $Time_i$ and $DollarCost_i$ are obtained by routing the tours by both generalized transit (*T*) and auto (*A*).

The target values for the chosen stratum, work tours, home location in very-urban zone, primary anchor location in very-urban zone, home near transit and primary anchor near transit, ONDN.w.1.1, are shown in Table 7.

Table 7. Target mode split from survey data.

Fraction	Anchor Mode
.39	Walk
.27	Transit
.34	Auto

2.2.1.2 Activity File Preparation

For calibration purposes, an activity file is created containing only the activities that are part of the 99 selected tours. Matching this is a one-traveler-per-household population file having only travelers whose tours were selected. The resulting activity file is slightly modified. Activity IDs are renumbered so that the N-trip tour is numbered from 1 to N. The upper and lower bound of the start time for the first activity of every tour is changed to zero, and the duration of that activity has the duration lower bound set equal to the end-time lower bound, and the duration upper bound set equal to the end-time upper bound. The final activity in the tour also has its times changed. The upper and lower bound of the end time for the last activity of every tour is changed to 27 hours, and the duration of that last activity has the duration lower bound set to the time difference of 27 hours and the original start time upper bound, and the duration upper bound set to time between 27 hours and the original start time lower bound. The population file is generated in a similar way. The household and traveler IDs are matched to those in the original population file. Only the household or person entries that match are included in the new population file. The field of the household lines that specifies the number of persons was changed to one for each household.

To match the target value definitions, subtours are eliminated from the mode calibration. These could be physically removed from the activity set, but here subtour activities have their modes changed to inter-household shared rides, or mode=8 in the Study. This particular mode is not simulated, and the Route Planner does not alter the departure or arrival times of the activity. This leaves the Route Planner's estimated travel times for the primary tour unchanged. The subtour mode change is accomplished by creating a new iteration database for the selected and re-numbered tours and travelers. That information is used to identify subtours. The modes in the subtours are changed to mode=8 through the Activity Regenerator. An "MS 8" Activity Regenerator command is prepared for each subtour, and the Activity Regenerator is executed. The resulting new activities are merged into the original activities. More information on Activity Regenerator feedback commands can be found in TRANSIMS Ver. 3, Volume Three (*Modules*), Chapter Three (*Activity Generator*), Section 5 (*Activity Regenerator*).

2.2.1.3 Obtaining a Cost Database and the Route Planner Proportions

The selected tours are routed by both modes *A* and *T* to determine the "costs" for the tours. Iteration and tour databases are created for both routings. These results are combined to make a tour-based cost database. The "rational choice" Route Planner mode split between all walk legs and at least one transit leg is determined from the routing data. These are used in conjunction with the target values to determine the target proportions for auto mode within the context of a given router transit mode. These proportions are used in computing Equations 14, 15, and 16 of Section 2.1 of the **GMD**. In the Study, light rail is combined with any transit, so the three equations become two where equation 15* is a combination of 15 and 16, with rail and bus treated together as "transit" (*t)*. That is, $b_t = b_c + l_c$, and Equation 15* is:

$$P_U^t(M_2 = A \mid s_t, D) = 1 - \frac{b_c + l_c}{(a_c + w_c + b_c + l_c)(p_R^b + p_R^l)} = 1 - \frac{b_t}{(a_c + w_c + b_t)p_R^t}$$

To route all of the sampled tours by transit: auto, walk, and rail mode trips are changed to the general transit (*T*) mode, which is mode= 3 in the Study. This step is completed using the "M 3 1 2 4" feedback command to the Activity Regenerator for every tour in the sample. Because all tours are affected, no merging is necessary.

The activities generated above are routed. Those routes are combined with the activities to create a transit iteration database. This database is converted to a tour-based database.

For the 99 samples, the Route Planner "rational choice" mode split between all walk legs and any transit leg results in 43 samples being walk, and the remaining 56 transit. These are summarized in Table 8.

Table 8. Route Planner generalized transit mode split.

Anchor Mode	Fraction
Walk	0.434
Transit	0.566

The values of Equations 14 and 15* are in Table 9. These are the calibration target proportions of auto assignments conditional on the "rational choice" behavior of the Route Planner. That is, if the Route Planner assigns a tour as walk when the assigned mode is generalized transit (*T*), then 90% of these tours should be assigned the mode *T*, while the remaining 10% should be assigned auto (*A*). When the Route Planner "rational choice" has at least one transit leg, the mode assignment is 48% generalized transit (*T*), and 52% auto (*A*).

Table 9. Calibration target proportions for Auto.

Anchor Mode	Proportion
Walk	Eq. 14 = 0.100
Transit	Eq. 15* =0.520

The transit tour database is used to create an activity feedback file where all generalized transit (*T*) trips are converted to auto (*A*). The feedback commands used for this purpose are of the form "M 2 3". The Activity Regenerator is executed with these commands to create an activity set where the tour modes are auto (*A*). The resulting activities have only auto and inter-household shared ride modes. These auto activities are routed, and the routes and activities are collated to create the auto iteration database. The iteration database is condensed into a tour-based form, and that database is further filtered to obtain the auto travel times for each tour.

The two tour-based databases—one for routing by auto, the other from routing by generalized transit—are combined to form a tour cost database. This file contains all of the information necessary to compute the generalized cost functions for each tour under both modes.

The fields of this database are given in Table 10. Fields labeled with a "T-" refer to the values for the generalized transit (*T*) mode routing. Those marked with "A-" refer to the auto (*A*) mode routing.

Table 10. Tour cost database fields.

Field	Description
HH	Traveler's HouseHold ID
Trav	Traveler ID
Act	Activity ID for the first non-home activity in the tour
Income	Household income in dollars
Parking	Zone ID of the parking location of the primary anchor activity
T-Time	Tour travel time, excluding subtours
T-AnchorMode	Mode of trip to anchor from previous activity
Distance	Accumulated Euclidean distance between activities on tour
A-Time	Tour travel time excluding subtours
A-AnchorMode	Mode of trip to anchor from previous activity
ParkingCost	Dollar cost to park in Anchor Parking on a work tour

The two calibrations, walk against auto and transit against auto, are performed separately. For the walk/auto calibration, an α of 0.0000033 yields the target 39% total walk tours (90% of the 43 walk-possible tours) as walks. For transit/auto calibration, an α of 0.00004 yields the target transit tours at 27% of total tours (or 48% of the 56 transit-possible tours) as transits. All remaining tours are auto, and therefore match the 33% target proportion of total tours. The proportions quoted here are for a deterministic evaluation of the cost function, obtained by choosing the mode with the lower cost for each tour. The results are summarized in Table 11.

Table 11. Calibrated mode split for the sample data.

Count	Anchor Mode
39	Walk
27	Transit
33	Auto

The purpose of the calibration stage is to determine the calibration parameters. In the Study, they were found to be α=0.0000033 to calibrate walk vs. auto tours and α=0.00004 to maintain calibration in tours with transit legs vs. auto tours. These calibration parameters are used in the Section 2.2.2 (*Application*).

2.2.2 Application

In actual applications, the parameters determined in the calibration stage are used to assign the modes for each tour in a forecast year. If all of the tours in the forecast year are routed by both generalized transit (*T*) and auto (*A*), then application proceeds by comparing the costs for the two routings and making a generalized transit (*T*) or auto (*A*) decision based on these costs. The decision may be made probabilistically using either a logit or a non-logit form of the calibration function. In the non-logit case, randomness may be introduced based on the costs nearest the tours in question. This procedure is

outlined in the **GMD**. When using a non-logit procedure, a pre-application sample of tours is drawn for each of the strata considered. These are routed using both modes—auto (*A*), and generalized transit (*T*). From these, the Route Planner's "rational choice" splits are determined. The data in each "rational choice" classification are used to establish the mode choice probabilities.

As an illustration of the application procedures in the Study, the base year is treated as the forecast year and a sample of 100 tours from the ONDN w.1.1 strata is chosen, and the modes are assigned to these tours. As a demonstration, modes are assigned to the tours in the sample rather than the entire set of tours in the forecast year ONDN w.1.1 strata. The three methods of mode assignment described Section 2.3.3 of the **GMD** are shown.

This Section is broken into five subsections, one describing the pre-application sampling procedure, and one illustrating each of the non-logit application procedures. The last subsection compares the results of the three methodologies and describes the differences between the pre-application and calibration samples.

In the second subsection, modes are assigned considering information obtained by routing by both auto and generalized transit. The third subsection shows the procedures when routing is available for only the generalized transit mode. In the fourth subsection, the technique for assigning modes using information determined by routing only auto tours is demonstrated.

2.2.2.1 Preapplication Sampling

A random sample of 100 tours is drawn from the ONDN w.1.1 strata and is used as the basis for the mode methodology applications. These tours are routed with both generalized transit (*T*) and auto (*A*) modes. The tours are split by the "rational choice" behavior of the Route Planner forming a set of walk and a set of transit mode choice data. The costs for auto and transit are evaluated with the calibration constants determined in the calibration phase. Fig. 3 shows the application sample cost data for both of the "rational choice" splits of the Route Planner. The calibration functions are superimposed on these plots as straight lines.

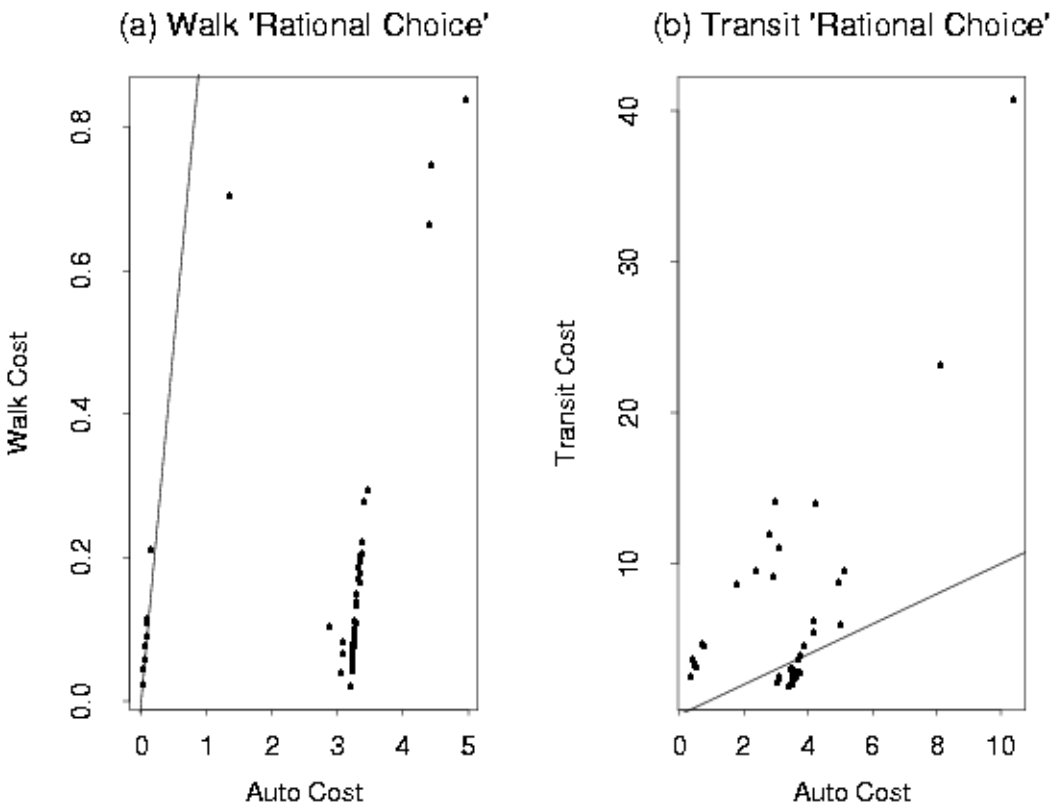

Fig. 3. Panel (a) shows the auto and walk costs for the application sample when the Route Planner "rational choice" mode is walk. The application costs for transit and auto when the Route Planner "rational choice" is transit are shown in panel (b).

The results of the Route Planner's split of the application sample into all walk legs and those requiring at least one transit leg are shown is Table 12. The differences in these proportions and those given in Table 8 for the calibration sample are discussed in Section 2.2.2.5.

Table 12. The proportions of application samples from the "rational choice" split from the Route Planner.

"Rational Choice"	Proportion of Sample
Walk	0.51
Transit	0.49

Table 13 shows the mode split for the 100 application samples obtained by a direct comparison of the auto and transit or walk costs. These deterministic counts are used as a comparison for the results of stochastic procedures given in the following sections. The mode splits for the application sample differ from the mode splits of the calibration sample given in Table 11. These differences are discussed in Sections 2.2.2.5 and 3.3.

Table 13. Deterministic mode split for the application sample.

Count	Anchor Mode
45	Walk
26	Transit
29	Auto

2.2.2.2 Mode Assignment Using Generalized Transit and Auto Costs

Assigning modes using both the auto and general transit costs follows the procedures outlined in Section 2.3.3 of the **GMD** and illustrated by panel (a) of Fig 3 in that document. In this methodology, each tour in the forecast year is routed twice. One route is completed using the auto mode (*A*), while the other is by generalized transit (*T*). Since each tour is routed by both modes, the "rational choice" selection of the Route Planner is known for the individual tours. Mode assignment for a particular tour is made by using a subset of the assignment sample appropriate for the "rational choice" selection of the Route Planner for that tour. The probability of assigning auto to the tour is determined from the proportion of tours with a lower auto cost than transit cost from the tours of the application sample with transit and auto costs similar to the transit and auto cost of the tour under consideration.

In the Study, the methodology is applied to each of the tours in the application sample. The average or expected values for multiple applications of the methodology to the application sample are given in Table 14. The deterministic mode splits given in Table 13 are reproduced here for comparison. Since the procedure is stochastic, multiple assignments of modes to the application sample give slightly different results. Variability of this and the other mode choice procedures is discussed in Section 3.

Table 14. The expected and deterministic mode split for the methodology using both generalized transit (*T*) and auto (*A*) costs to the application sample.

Expected	Deterministic	Anchor Mode
44.8	45	Walk
26.6	26	Transit
28.6	29	Auto

The final step in the methodology is to iterate to remove unreasonable mode choices. In this step, walks that are considered to be too long or transit tours with legs requiring too many transfers are changed to auto tours. For each mode change to auto, a corresponding auto tour is stochastically selected and its mode changed to the appropriate mode. This step is not completed in the Study because the stochastic selection requires that all the tours in the ONDN w.1.1 strata have mode assignments.

The need to iterate is not unique to only the non-logit methods like the one used in the Study. Each mode assignment methodology, whether it is logits or some other methodology, is based on aggregate statistics rather than particular individuals. As a consequence, a small percentage of individuals will have unreasonable tour modes independent of the mode assignment methodology.

2.2.2.3 Mode Assignment Using Generalized Transit Costs Only

The second assignment methodology considers only the generalized transit cost, and the mode choice is made independent of the auto costs. The procedure is illustrated in panel (b) of Fig 3 in Section 2.3.3 of the **GMD.** Since the routing in this assignment methodology is by generalized transit, the "rational choice" selection of the Route Planner is known. The probability of assigning auto to the tour is determined by identifying points in the application sample that have transit costs similar to the one under consideration. The proportion of tours with a lower auto cost is computed for these "nearby" samples and is taken to be the probability of assigning auto to the tour.

The average or expected values of the mode splits resulting from using this procedure multiple times on the application sample are given in Table 15. As before, this methodology is stochastic and repeated applications of the method produce a distribution of results. This variability and a comparison with the results for the other methodologies is given in Section 3.

Table 15. The expected and deterministic mode split for the application data using only generalized transit costs.

Expected	Deterministic	Anchor Mode
45.4	45	Walk
26.2	26	Transit
28.4	29	Auto

2.2.2.4 Mode Assignment Using Auto Costs Only

The final sampling method demonstrated here is the one that is conditioned on auto costs only. Since the tours are not routed using generalized transit, the "rational choice" behavior of the Route Planner is unknown. In this case, the probability of a particular generalized transit selection—all walk legs or at least one transit leg in the Study—is estimated from the auto costs, the "rational choice" probabilities from the Route Planner for the application sample, or from a surrogate measure such as distance traveled. For each tour in the forecast year, the "rational choice" grouping is selected stochastically using these probabilities. After the grouping is chosen, the procedure is similar to the choice selection when the transit costs alone are known, with the auto costs taking the place of the transit costs. These procedures are documented in Section 2.3.3 of the **GMD**.

Routing by generalized transit is not completed for the tours in the forecast year. Therefore, the tour must be randomly assigned either the all walk or at least one transit leg sets, s_w and s_t, by a mathematically justified method. If the probability of being in one or another of these two groups is determined to be independent of the auto costs or any surrogate of the auto costs, then the values in Table 12 can be used as estimates of the assignment probabilities. That is, $p_w = .51$ and $p_t = .49$. If, on the other hand, the probability of being in one of these two sets is not independent of the auto costs, then a method has to be devised that takes into account these dependencies.

This lack of independence is the case in the Study as the probabilities of having all walk legs or having at least one transit leg are a function of the distance traveled. Figure 4 shows probability density estimates of the travel distances for the application sample for the two sets of data—those tours that the Route Planner places in the transit set and those in the walk set. It is apparent in this figure that the longer the distance the more likely it is that the Route Planner assigns transit to the tour. Therefore, the probabilities that the Route Planner assigns the transit or walk set are estimated as a function of distance.

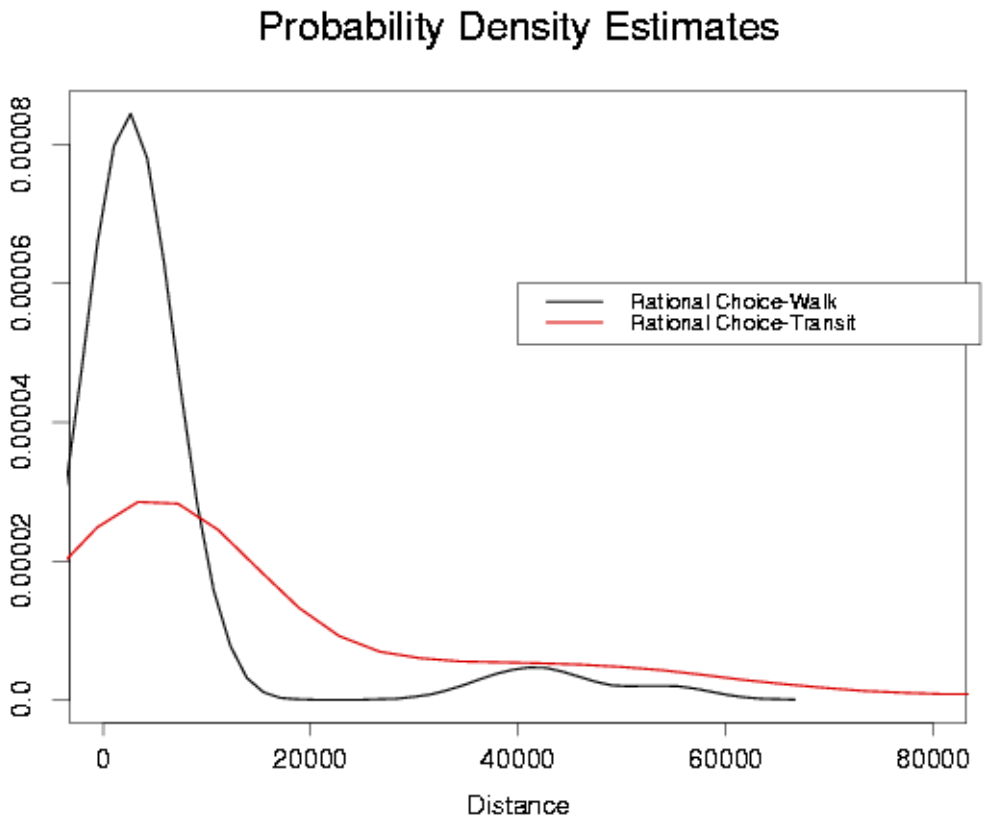

Fig. 4. Probability density plots of the distances from the application sample for the walk and transit application data. The black line is the density function for distance from the walk tours. The red line is for the transit tours.

A simple logit based on the distance traveled and the Route Planner "rational choice" in the application sample is used to estimate the probabilities of the Route Planner selection. The estimated probability of at least one transit leg as a function of distance is:

$$P(s_t) = 1/(1+\exp(.464 - .0000422 * \text{distance}))$$

Mode assignment is made in two steps. For each application sample, the "rational choice" selection of the Route Planner is determined stochastically using the tour distance and the logit given above. Depending on that choice—either auto and walk set, or the auto and transit set—the mode choice of auto is established randomly as described in Section 3.2.2

of the **GMD**. The expected value of the resulting mode split for the application sample is shown in Table 16.

Table 16. The expected and deterministic mode split for the application sample using only auto costs and a simplified logit to estimate the "rational choice" behavior of the Route Planner.

Expected	Deterministic	Anchor Mode
44.8	45	Walk
25.9	26	Transit
29.3	29	Auto

If the mode split using the auto costs only is made employing only the mode split proportions $p_w = .51$ and $p_t = .49$ in place of the logit to determine the choice of using the walk or transit calibration function, the mode assignment results are biased. The expected mode split for this case is given in Table 17. From this table there is an apparent bias in the number of transit and auto tours. This bias is the result of the lack of independence between the auto costs (or travel distances) and the Route Planner's "rational choices".

Table 17. The expected and deterministic mode split for the application sample using only auto costs and the "rational choice" mode splits of the application sample to estimate the "rational choice" behavior of the Route Planner.

Expected	Deterministic	Anchor Mode
44.7	45	Walk
29.4	26	Transit
25.9	29	Auto

2.2.2.5 Comparison of the Results

The procedures illustrated above are based on relatively small samples—99 samples for the calibration set, and 100 for the application set. These small sample sizes lead to variability in the final mode splits. The purpose of this subsection is to compare the results of the mode splits in the calibration sample and the three applications to determine if the differences in these results are within the variability expected for samples of size 99 and 100.

The "rational choice" mode split of the Route Planner for the calibration and application samples are given in Table 8 and Table 12 as proportions of the sample sizes. The numbers of walk or transit Route Planner choices for the two samples of size 99 and 100 are given in Table 18.

Table 18. Counts of the "rational choices" made by the Route Planner for the 99 calibration and 100 application samples.

Calibration Sample	Application Sample	"Rational Choice"
43	51	Walk
56	49	Transit

At first glance, splits in Table 18 seem to be divergent, but the contingency χ^2 for these counts is 1.14 with 1 degree of freedom and a P-Value of .19. Thus the results in Table 18 are will within the variability expected with samples of size 99 and 100.

The resulting expected mode splits for the three application methods are given in Tables 14, 15, and 16. They are recapped in Table 19.

Table 19. The expected and the deterministic mode splits using the three methods for mode assignment.

Deterministic	Using Auto and Transit Costs	Using Transit Costs Only	Using Auto Costs Only and the Logit	Using Auto Costs Only and the "Rational Choice" Probabilities	Mode
45	44.8	45.4	44.8	44.7	Walk
26	26.6	25.9	25.9	29.4	Transit
29	28.6	28.4	29.3	25.9	Auto

From Table 19, all of the methods except the auto costs only using the application sample "rational choice" probabilities, $p_w = .51$ and $p_t = .49$ give unbiased mode splits.

The variability in the applications of these methods is discussed in Section 3.3.

3. SOME TOPICS OF INTEREST

The procedures for mode assignment given in this document are the result of a general research project to understand mode choice in the context of the TRANSIMS framework. The non-logit form of the utility functions used in the Study is itself a research project to ascertain whether these simpler semi-parametric models can be used as mode choice models in place of the often more complex logit models now in use. Throughout the Study, numerous techniques were tested and some of them either failed or required slight modifications for the mode techniques to be successful. This Section contains some constructive information about these modifications and their effects on mode choice procedures.

3.1 Cost Functions That Can Not Be Calibrated

The basic form of the cost function used in the Study is the result of an assessment of those factors that influence mode choice. The first attempt at calibration did not include the $.03 per km cost for auto tours. In Table 5, only 29 out of 1260 TAZs have nonzero associated parking costs. If the $.03 per km driven is removed from the *DollarCost* portion of the cost function, then the *DollarCost* for auto tours is the parking cost alone. In the numerous drive tours where there is no parking cost, the comparison of the cost functions for auto and generalized transit becomes a comparison of the travel times by each mode. Since travel by auto is generally faster than by transit, auto is the preferred mode of travel in these cases. The over abundance of preferred driving tours leads to mode choices that can not be calibrated; that is, the transit calibration target value is not between the two horizontal lines in Fig. 2 showing the maximum and minimum possible proportion of transit tours.

3.2 Nonuniqueness of the Cost Function

Changing the value of α in the cost function changes the proportion of walk or transit tours in the walk/auto or transit/auto calibration. The proportion of walks in the calibration sample as a function of α is shown in Fig. 5. In this figure, the proportion of walks decreases as the weight, α, on the travel time and the household income increases. The calibration procedure is to find one value of α that places the proportion of the walk tours in calibration. The points in Fig. 5 form a decreasing step function. From this figure, it is apparent that multiple values of α produce the same proportion of walk tours. The steps in this function occur as each point in the calibration sample moves from one side of the line shown in Fig. 1 to the other as α changes. Therefore, a smoother function may be obtained by increasing the calibration sample size.

In the Study, an appropriate value of α is obtained by using a search technique. Another option is to compute the proportion for numerous values of α and fit a parametric function to the resulting data. For example, a least squares fit of the data in Fig. 5 could be made to a functional form such as:

$$p = \frac{\beta_0}{1 + \exp(\beta_1 \alpha)}$$

where β_0 and β_1 are parameters determined by the fit. The function would then be inverted and α determined for a particular value of *p*.

The proportion of walk tours as a function of α in the cost function used in the Study is non-increasing. The differences in travel times using the two modes and the differences in the corresponding dollar costs are in the same quadrant of a plot similar to that in Fig. 1. Therefore, the particular cost function used in the Study does not exhibit non-monotone behavior.

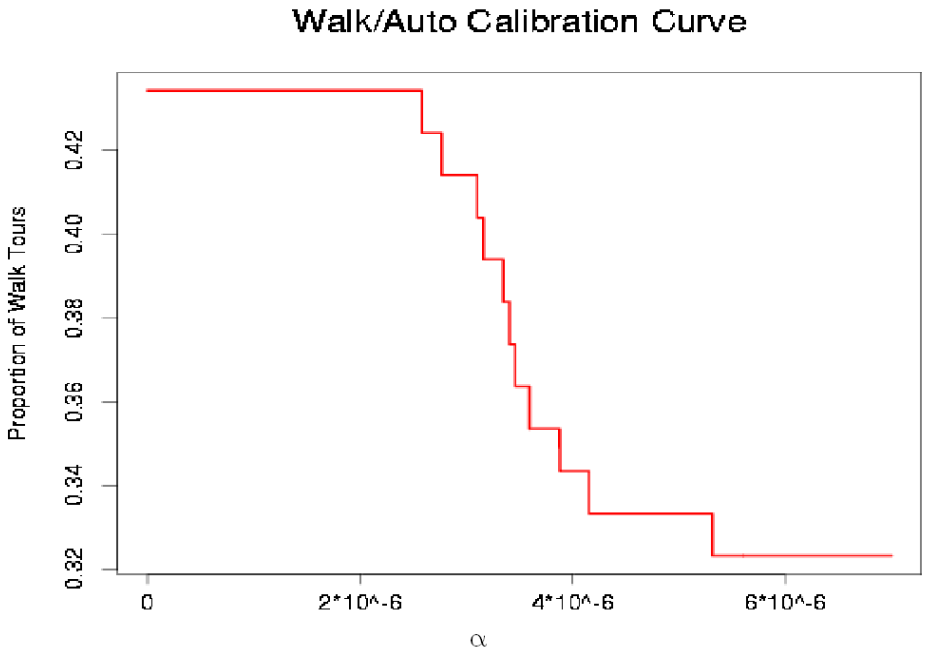

Fig. 5. Calibration curve of walk fraction versus the calibration parameter.

3.3 Sample Sizes, Biases and Variability

The expected values given in Section 2.2 do not address the variability inherent in the recommended methods. The purpose of this section is to consider the variability and biases of these methodologies. The variability, precision and biases in the three methods for mode assignment depend on the sample sizes used in all stages of the methodologies.

In the Study, the Route Planner "rational choice" proportions varied greatly between the calibration and application sample. This amount of variability seen there is within the

variability expected for samples of size 99 or 100. Differences this large in mode splits between the calibrations and an application sample for the base year may be unacceptable in actual applications. Increasing the sample sizes until the desired variability is obtained can control the differences in the mode splits. As rule of thumb the standard deviation of any estimated proportion is no more than $\frac{1}{2\sqrt{n}}$, where n is the sample size. If the analyst wanted the estimated percentage of one of the mode splits to be within, say, .05 of the true value (that is 2 standard deviations are within .05 of the true value), then the sample size for the application and calibration samples would be $n = 1/(.05)^2 = 400$. In almost all cases, sample sizes of 500 to 1000 should be ample to calibrate the system.

In the Study, the five nearest application points were used to determine the expected values for the mode split methodologies. As shown in Table 19, this resulted in a slight bias away from the deterministic results for the application sample. If the procedure is changed to include either the nearest 7 or 10 points, this bias increases. These results are given in the Table 20.

Table 20. The changes in the expected results for the methodologies as the nearest "number**" of points are used to estimate the probabilities.**

walk	transit	auto	number	application type
45.0	26.0	29.0		deterministic
44.8	25.9	29.3	5	Auto Only Using Distance Logit
43.8	27.7	28.5	7	Auto Only Using Distance Logit
44.9	30.0	25.2	10	Auto Only Using Distance Logit
44.7	29.4	25.9	5	Auto Only Using Estimated Proportions
43.8	30.6	25.6	7	Auto Only Using Estimated Proportions
44.6	32.7	22.7	10	Auto Only Using Estimated Proportions
45.4	26.2	28.4	5	Transit Only
44.4	28.0	27.6	7	Transit Only
43.8	29.1	27.1	10	Transit Only
44.8	26.6	28.6	5	Transit and Auto
44.4	28.0	27.6	7	Transit and Auto
45.0	30.0	25.0	10	Transit and Auto

In Table 20, the column "**number**" is the number of nearest neighbors used to estimate the choice probabilities. The difference between the deterministic results and the stochastic results increases as the number of nearest points increases. In the Study, using the five nearest points produced the most reliable results.

The actual number of nearest points to use in the application depends on the original sample size. It is recommended that, in actual applications using the non-logit methods described here, a similar experiment be performed using the calibration sample to determine the appropriate number of nearest points to use.

In addition to the expected values of the application of the methodologies, the variability in the methods may also be computed. The standard deviation of the application sample mode splits for each of the methodologies and estimation number is given in Table 21.

Table 21. The standard deviations of the estimated mode splits of the application sample.

walk	transit	auto	number	application type
4.6	4.0	2.9	5	Auto Only Using Distance Logit
4.5	4.2	2.9	7	Auto Only Using Distance Logit
4.6	4.3	3.0	10	Auto Only Using Distance Logit
4.8	4.0	3.1	5	Auto Only Using Estimated Proportions
4.8	4.1	3.0	7	Auto Only Using Estimated Proportions
4.8	4.3	3.1	10	Auto Only Using Estimated Proportions
2.0	1.6	2.6	5	Transit Only
2.3	1.7	2.8	7	Transit Only
2.5	1.7	3.0	10	Transit Only
1.5	1.3	2.0	5	Transit and Auto
1.5	1.5	2.1	7	Transit and Auto
1.5	1.8	2.4	10	Transit and Auto

The entries in Table 21 are interesting. The variability in the process decreases as more information is used to determine the mode choice. For example, consider the walk mode split. When only auto costs are used to determine the walk mode split, there is no information other than auto cost or distance traveled to determine the "rational choice" behavior of the Route Planner. As a consequence of this, the standard deviation of the walk mode split for the 100 application samples is around 4.6. This is in contrast to the 2.0 to 2.5 standard deviation when the transit cost, and hence the 'rational choice" of the Route Planner, is known. The smallest variability is obtained when the tour is routed both by auto and generalized transit where the standard deviation is 1.5.

3.4 Boarding Times for Transit

The TRANSIMS Route Planner finds the least time path through the walk and transit network when the mode is generalized transit. When aggregated, the collection of paths produces a model for the behavior of a transit population. The form of this model is dependent on the transit schedules and routes, the representation of transit stops, and input parameters that control the time a traveler must be at the transit stop before the scheduled departure. The two configuration keys that govern this are

`ROUTER_GET_ON_TRANSIT_DELAY` and `ROUTER_GET_OFF_TRANSIT_DELAY`. The names of these two parameters should not be taken literally. They should be viewed as penalties in making transfers due to the inconvenience of the transfer and the uncertainty in the transit schedules.

Setting the values of these delay times to well over one minute produces better models of the behavior of transit travelers. That is, under these penalties, travelers are less likely to choose routes that require multiple transfers with short legs since there is a time penalty for each transfer. A discussion of the resulting transit behaviors and the recommended delays may be found in the modeling document, TRANSIMS Ver. 3.0, Volume Eight.

The use of the delay parameters as penalties not only produces correct transit transfer behavior, but also affects the mode choice calibration. In the Study, if the boarding delay for transit is set to one minute or less for the calibration sample, the calibration between walks and autos cannot be set. If the boarding penalty is low and there is an abundance of transit possibilities, such as the bus mall in Portland, then the Route Planner will not mimic the behavior of most travelers. In these cases, a short transit trip of one or two transit stops becomes more favorable than walking between the destinations. When this happens in the calibration stage, the Route Planner (in its "rational choice") places too many travelers on transit and has too few walking. In the Study, this leads to situations where calibration of the walks is impossible.

Table 22 contains the values of Equations 14 and 15* for the calibration sample when a small transit delay time is used for routing. Equation 14 is negative and from the **GMD** negative values for these equations indicates a situation where calibration is not possible. Examination of the form of Equation 14 in the **GMD** shows that it will become negative when the proportion of Route Planner "rational choices" for one of the alternatives becomes too small, which is the case here for the walk "rational choice".

Table 22. Calibration target proportions for Auto with small boarding time delays.

Anchor Mode	Proportion
Walk	Eq. 14 = -0.39
Transit	Eq. 15* =0.62

4. SUMMARY

This document demonstrates multiple methodologies for mode assignments using the TRANSIMS framework. Mode assignment in the Study is a research project. First, the implications of the "rational choice" behavior of the Route Planner are documented here. This behavior requires special procedures for calibration. These procedures are given in the **GMD** and their application is shown here using a small example.

In the second research effort, the use of non-logit calibration functions is demonstrated. It is shown with a small example that simple calibration functions for small subsets of the population may be used in place of the logit functions commonly used on larger subsets of the populations.

For the non-logit case, three methodologies for mode assignment to the forecast year are given. It is shown that each methodology produces an unbiased mode assignment. However, the methodology using only the auto costs is made unbiased by estimating the "rational choice" behavior of the Route Planner employing a simple logit function of the distance traveled. The variability of these methodologies is shown to increase as the amount of information used to make the mode assignment decreases.

Being a research project, numerous techniques for mode assignment were investigated throughout the Study. Many of these failed or had to be modified to obtain unbiased mode splits. Most notably, a transfer penalty for light rail and bus tours had to be added to produce realistic transit behavior. This penalty is recommended for all TRANSIMS applications. Additionally, a cost for the distance traveled had to be added to the generalized cost function for auto tours to allow for calibration.

A set of general procedures for mode choice is given in Appendix A of this document. The scripts and configuration files used in the Study are in Appendix B. A general set of scripts and configuration files are in Volume Seven (*Appendix*), Chapter Thirteen (*Mode Feedback*) in the set of Study documents

5. Appendix A: General Scripts for Mode Calibration

This appendix contains a description of a general set of scripts that may be used for mode calibration and application to a full set of tours for a forecast year. The scripts are found in Volume Seven (*Appendix*), Chapter Thirteen (*Mode Feedback*) of the Study documents.

5.1 Calibration Methods

The purpose of the calibration step is to determine the model parameters from the base-year data. Only the model parameters from this procedure are retained for the forecast year. All other calculations and data are discarded. In this methodology, the travel tours of the population are divided into strata. Tours are stratified by the type of trip and other characteristics that may influence mode choice, such as the primary anchor of the tour—work, shop, or other—and the distance to transit. There are two steps to the calibration.

1) The first step is the generation of the data necessary for the calculations, called "initialization". This step involves determination of the base-year target values for mode splits from a source outside of TRANSIMS, like survey data. The TRANSIMS data necessary for calibration is generated from a random sample of tours from each of the strata. These tours must be routed on both auto and generalized transit modes.

2) The second step is the set of calculations that determine the model parameters based on data from the initialization step.

5.1.1 Initialization

5.1.1.1 Select Tour Types

Only the strata with home and primary anchor locations near transit, ONDN, is described in this document. Further stratification of the tours in the ONDN group is by work and non-work tours, home urbanization, and primary anchor urbanization.

All of the analysis for mode feedback is done with tour-based databases. Since the Activity Regenerator and the Route Planner both produce output by household, every database is filtered so that only ONDN tours are included. Further, since each tour must be considered independent of all other traveler's tours, each tour modification is considered separately. That is, the entire procedure is done once for each tour, with all other tours of that traveler in their original state each time. For a traveler with five tours selected for testing, the entire initialization procedure is done five times. When the data for all selected traveler-tours is available, the databases are merged so that all selected tours are considered together.

Fig. 6. Creation of the AS7-ONDN tour database.

Fig. 6 shows the creation of the initial ONDN tour-based database from the AS7 set of activities. All of the flowcharts shown as figures in this section present the progress of files (black) and scripts and programs (blue) run as part of mode feedback. Here, the TRANSIMS Collator is used to collate the AS7 activity data with the network and population data. This process is run in parallel by a *RunCollator* script. The subsequent output is merged into a single database. Merging is accomplished by concatenating all other databases to the first file, excluding their headers. This iteration database has one entry for each activity in AS7. It is used as input to a perl script that condenses the information into a tour-based format described in Section 3.1.3. The tour database contains one entry for every tour in AS7, so it is then filtered through a second perl script that removes all entries not in the ONDN group. Also in this script, all travelers below the age of 16 are removed. The result is the initial ONDN tour database based on AS7 activities.

The resulting ONDN database is stratified into 18 separate strata. These strata were determined and subdivided again according to transit type "rational choice" as determined by the Route Planner. Each of these 54 groups will be calibrated independently. This is done most efficiently in the procedure described in Section 5.1.1.3.

5.1.1.2 Compute Targets

The base-year target mode splits are calculated using only survey data. These numbers are employed in Equations 14, 15, and 16 of Section 2 to determine the calibration.

The starting point is the set of all trips from the two-day activity survey, having 73,265 entries. The data is not adjusted to account for sampling bias in the survey. Excluding any traveler under the age of 16, the set reduces to 59,756 trips. Excluding all trips that belong to a subtour, there are 56,886 remaining trips. All trips on any tour that have any leg done as an auto passenger or as a driver with a passenger are also removed, reducing the set to 28,171 trips.

The set of trips is then converted to a set of tours. All activities are classified as work or non-work activities. If there are any work activities in a tour, only the work activity with the longest duration in that tour is kept in the data set. If there are no work activities, only the activity in the tour with the longest duration is kept in the data set. The duration is the time spent at a single activity location. The resulting database of trips representing tours has 11,641 entries, although Table 23 reports only 9,780.

Each tour is classified according to the stratification criteria described in Section 3.1. If the distance of the home location to transit is less that 1000 meters, the tour is said to be "origin near transit". The primary anchor location is selected in the above procedure. If this activity location is within 1000 meters of any transit stop, the tour is said to be "destination near transit". The final two classifications refer to the urbanization value of

the home location and of the primary anchor location. The possible values are "suburban", "urban", or "very urban".

Table 23. Calibration target mode splits determined from the survey.

work/non-work	home urbanization	work urbanization	transit distance	Walk	Bike	Transit	Park & Ride	Drive Alone	Total
work	very-urban	very-urban	ONDN	59	2	41	0	51	153
work	very-urban	very-urban	OFDN	0	0	0	0	0	0
work	very-urban	very-urban	DF	0	0	0	0	0	0
work	very-urban	urban	ONDN	14	3	2	0	39	58
work	very-urban	urban	OFDN	0	0	0	0	0	0
work	very-urban	urban	DF	0	0	0	0	0	0
work	very-urban	suburban	ONDN	0	2	9	1	52	64
work	very-urban	suburban	OFDN	0	0	0	0	0	0
work	very-urban	suburban	DF	0	0	0	0	0	0
work	urban	very-urban	ONDN	44	24	101	6	242	417
work	urban	very-urban	OFDN	0	0	0	0	0	0
work	urban	very-urban	DF	0	0	0	0	0	0
work	urban	urban	ONDN	23	13	31	0	256	323
work	urban	urban	OFDN	0	0	0	0	0	0
work	urban	urban	DF	0	0	0	0	0	0
work	urban	suburban	ONDN	7	2	21	0	430	460
work	urban	suburban	OFDN	0	0	0	0	0	0
work	urban	suburban	DF	1	2	0	0	41	44
work	suburban	very-urban	ONDN	33	8	60	140	414	655
work	suburban	very-urban	OFDN	1	0	2	25	54	82
work	suburban	very-urban	DF	0	0	0	0	0	0
work	suburban	urban	ONDN	10	4	12	7	372	405
work	suburban	urban	OFDN	1	0	0	0	59	60
work	suburban	urban	DF	0	0	0	0	0	0
work	suburban	suburban	ONDN	65	19	43	12	2225	2364
work	suburban	suburban	OFDN	4	0	0	1	552	557
work	suburban	suburban	DF	17	3	0	0	380	400
non-work	very-urban	very-urban	ONDN	109	4	12	1	29	155
non-work	very-urban	very-urban	OFDN	0	0	0	0	0	0
non-work	very-urban	very-urban	DF	0	0	0	0	0	0
non-work	very-urban	urban	ONDN	13	5	7	0	26	51
non-work	very-urban	urban	OFDN	0	0	0	0	0	0
non-work	very-urban	urban	DF	0	0	0	0	0	0
non-work	very-urban	suburban	ONDN	4	0	8	0	13	25
non-work	very-urban	suburban	OFDN	0	0	0	0	0	0
non-work	very-urban	suburban	DF	0	0	1	0	1	2
non-work	urban	very-urban	ONDN	29	5	26	1	94	155
non-work	urban	very-urban	OFDN	0	0	0	0	0	0
non-work	urban	very-urban	DF	0	0	0	0	0	0
non-work	urban	urban	ONDN	138	16	21	0	295	470
non-work	urban	urban	OFDN	0	0	0	0	0	0
non-work	urban	urban	DF	0	0	0	0	0	0
non-work	urban	suburban	ONDN	16	2	9	0	170	197
non-work	urban	suburban	OFDN	0	0	0	0	0	0
non-work	urban	suburban	DF	1	0	0	0	11	12
non-work	suburban	very-urban	ONDN	11	0	12	8	73	104
non-work	suburban	very-urban	OFDN	1	0	0	1	17	19
non-work	suburban	very-urban	DF	0	0	0	0	0	0
non-work	suburban	urban	ONDN	14	1	4	0	200	219
non-work	suburban	urban	OFDN	0	0	0	0	14	14
non-work	suburban	urban	DF	0	0	0	0	0	0
non-work	suburban	suburban	ONDN	212	14	38	4	1527	1795
non-work	suburban	suburban	OFDN	1	0	0	0	244	245
non-work	suburban	suburban	DF	31	4	1	0	239	275
			Total	**828**	**129**	**460**	**207**	**7881**	**9780**

Table 23 shows the result of this analysis. The categories described above match those described in Section 3.1 and are used in the Study as calibration targets. A more careful generation of targets would account for the weighting of households to match the actual demographics of the population as determined from the census,.

Note that there are 25 classifications with no data, and 29 with data. It is assumed that the groups with no data are unpopulated. All of the ONDN groups have data, so this is not an issue in the calibration described in Section 5.1.2.2. However, there are several bins in ONDN with very little data, making the target mode split questionable. This and related topics will be discussed in the summary at the end of this report.

5.1.1.3 Route Samples

Part of the initialization step is to obtain the subset of TRANSIMS data intended for calibration and modeling. The first thing is that all walk and rail modes are converted to general transit, and the activities are regenerated with these modes. By so doing, all trips that are not bike, school bus, or magic moves become either auto or general transit, the starting point for mode feedback (mode *X*, as described in Section 2).

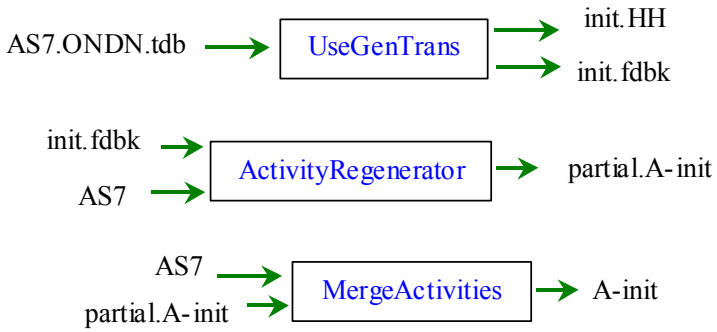

Fig. 7. Conversion of all walk and rail trips to general transit mode.

In the Study all walk and rail trips are converted to general transit. Fig. 7 describes the simple procedure. The ONDN AS7 tour database created in the previous section is used as input to the *UseGenTrans* perl script. For each tour in the database, a feedback command is created that converts all walk or rail modes in the tour into general transit. The activity feedback file is called *init.fdbk*. Every household with an entry in *init.fdbk* also has an entry in the household file *init.HH*, although this is not used. Since almost all households have at least one walk trip, *init.HH* contains almost all households.

The partial activities are then merged with the original set to recover any households that had no walk or rail trips. The result, *A-init*, includes all activities for every household. The *MergeActivities* step first creates activity household-indexes (if they are not already present) using the TRANSIMS program *IndexActivityFile*, then combines them using the *MergeIndices* program. Finally, *IndexDefrag* creates the defragmented activity file *A-init*. This activity file contains no walk, bus, or rail trips, only general transit.

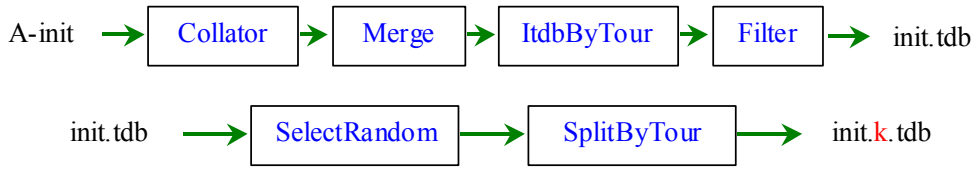

Fig. 8. Create initial tour database, select a subset of tours, then split into files with all first tours in k=1, all second tours in k=3, etc.

Because all mode feedback is done using tours, a tour database must be created from the initial activity set *A-init*. Fig. 8 describes this process. The first line of the figure is similar to Fig. 6: the *A-init* activities are collated with the network and population data in a set of independent parallel processes, their output is merged, the resulting file is input to *ItdbByTour*, and the result has all non-ONDN tours removed to form the final tour database *init.tdb*.

Since it is not necessary to test all tours to calibrate the model, only a fraction of the tours is selected for analysis. The size of this fraction is such that the least populated stratum will have at least 100 data points. The *Stratify* script used in Fig. 16 has an alternate mode of operation that does not stratify a database, but merely counts the occupancy of each bin. This should be used on *init.tdb* to determine the sampling fraction, but is not included in Fig. 8. Using the sampling fraction and *init.tdb*, a subset of tours is selected. However, as mentioned above, routing of different tours for the same traveler is done in separate runs of the Route Planner. The sampled set of tours is divided into several files, one for each tour number. The *init.1.tdb* file contains all selected tours that were the first tour of their respective travelers. The files, *init.k.tdb* are created for all necessary values of *k* where the red *k* is used to denote the tour number. The files are merged together at the end of the procedure. This merge is described in Fig. 13. All of the procedures from here to the last merge must be executed once for each of the values of the red *k*, from one through the maximum tour number.

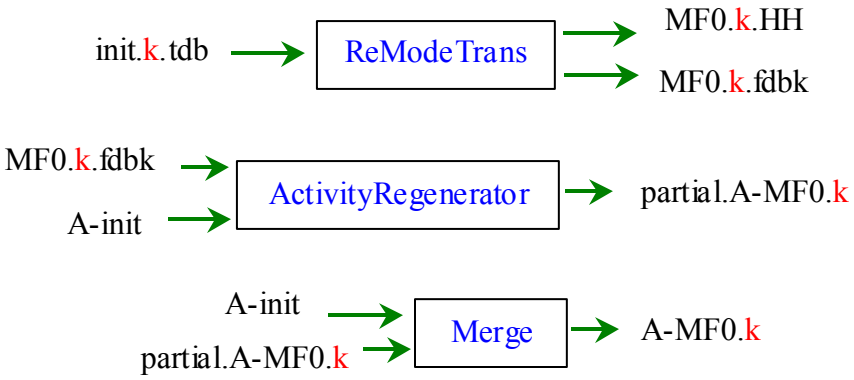

Fig. 9. Convert all selected tours to general transit mode.

The next step in the procedure is to route all selected tours on the general transit mode, denoted by *T* in preceding sections. This first involves generating a new activity set, as shown in Fig. 9. A new activity sets is generated for each value of *k*.

For each value of *k*, the database of selected tours, *init.k.tdb*, is used with the *ReModeTrans* script to create a set of feedback commands and household files similar to what was done with *UseGenTrans* in Fig. 7. For each auto mode tour in the input database, *ReModeTrans* creates a feedback command to convert the tour into transit mode tour. The feedback command issued is one of the *M* commands with optional arguments, described in the documentation for the Activity Generator. It converts any auto trip in the tour into a general transit trip, but leaves all other modes unchanged. The activity feedback file is called *MF0.k.fdbk*, for the zeroth mode feedback step. Each entry in the feedback file has a corresponding entry in the router feedback household file, *MF0.k.HH*.

The activity feedback file is used with the initial activities in the Activity Regenerator to create a partial activity file, *partial.A-MF0.k*, where all of the feedback commands have been executed. The partial activities are finally merged with the initial activities to form an activity set that is identical to *A-init* except that all auto mode tours are converted to transit mode tours. The resulting activity set is called *A-MF0.k*. Note again that each step is done for each value of *k*.

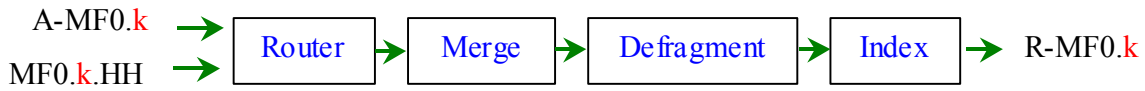

Fig. 10. Route selected tours on general transit mode.

The routing of the transit mode activities is shown in Fig. 10. The *A-MF0.k* activities are routed using the Route Planner in distributed-parallel fashion. The routing is carried out using a script called *RunRouter* that spawns a specified number of distributed parallel jobs of the TRANSIMS Route Planner. The resulting independent output files are merged using the *PlanFilter* program that operates on the plan indexes. The resulting combined index file is defragmented with *IndexDefrag*. The old indexes are removed, and the combined file is re-indexed for more efficient access. The result is a combined plan file called *R-MF0.k*. Note that the plan index is only one file, and its base-name is given as the argument for the plan file in the configuration file for the Collator. However, the "file" itself may be made up of many smaller fragment plan files. This is an important point that is explained in more detail in the Report on Route feedback. Note also that this must be done for each value of *k*.

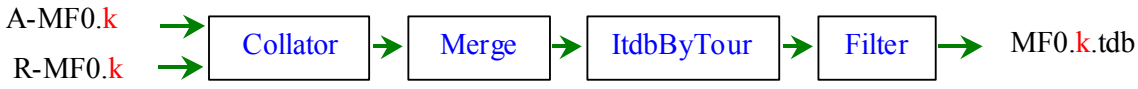

Fig. 11. Create tour database for transit mode tours.

With the activities routed, all data is available to create a new tour database with travel times taken from the routes. This is shown in Fig. 11. The procedure is the same as that described for Fig. 6, except that the file names as well as the results are different. By including the route table in the input, route times and modes are included in the database.

The resulting database of tours routed on transit is *MF0.k.tdb*, where this procedure is done once for each value of *k*.

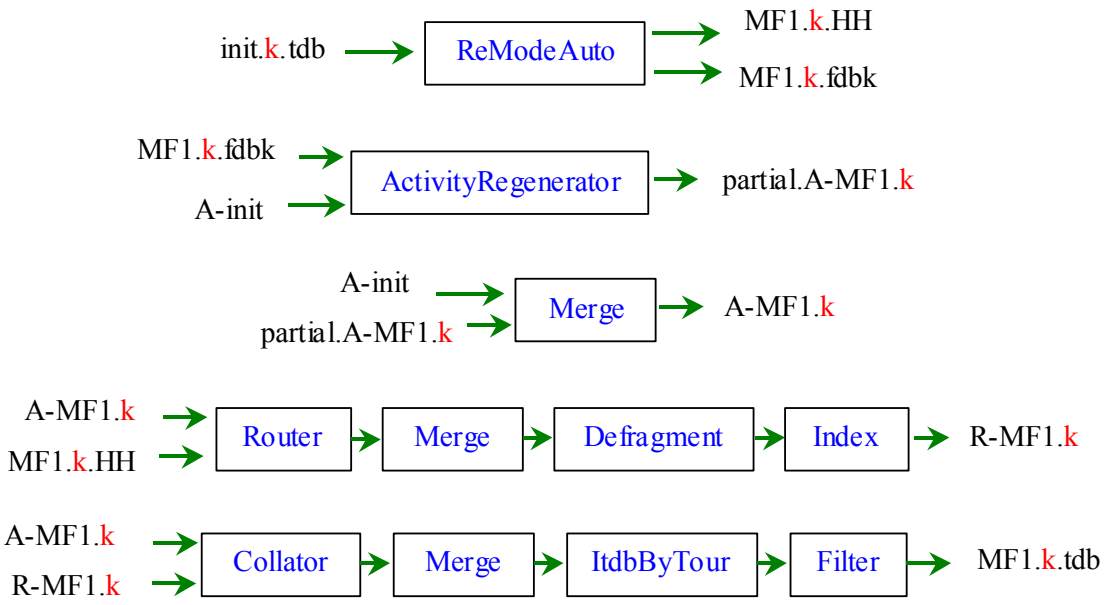

Fig. 12. Route selected tours on auto mode and create tour database.

The same method that created the transit mode database is used to create the auto mode database in Fig. 12. Starting with the initial tour database, *init.k.tdb*, feedback files are generated and used to change all transit mode tours to auto mode tours. The activities are routed, and both routes and activities are used to create a tour-based iteration database called *MF1.k.tdb*. As with all steps since Fig. 8, this is done once for each value of *k*.

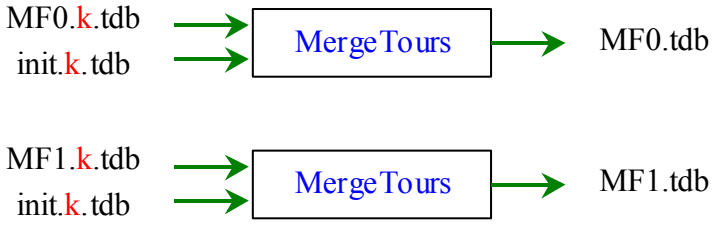

Fig. 13. Merge independent-tour databases into single ONDN databases.

The initial databases of tours selected for testing, *init.k.tdb*, contain entries only for the selected tours, and only one entry per traveler per file. On the other hand, the re-moded databases contain entries for all tours of all travelers in all households where any tour was selected. The same household may appear in databases with different values of *k*. It is, therefore, necessary to sort through all of these databases and collect only the tours that are being tested in each. This is done by the *MergeTours* script that takes all databases containing tours to be merged and uses the *init.k.tdb* files as the key to knowing which tours come from which files. The result is a file with entries for every tour tested, including the tested value of travel times and mode strings. As shown in Fig. 13, this is

done once for the transit tours and again for the auto mode tours, creating two tour-based databases as the output from the initialization step, *MF0.tdb* and *MF1.tdb*.

5.1.2 Calibration

This step determines the calibration parameters for the mode choice model. Although there are a large number of intermediate files, only the calibrated model parameters are retained for the forecast year.

5.1.2.1 Estimate Route Planner Probabilities

Using the perl script *CombineDB*, the two databases created in the last step of initialization are combined into a single database having all demographic, home, and primary anchor location information, and mode-independent tour data. Also included are the tour mode and travel time for each of the modes in *MF1.tdb* and *MF0.tdb*, which have auto in one and general transit in the other. This is shown graphically in Fig. 14.

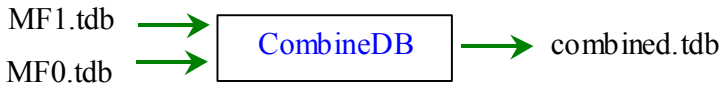

Fig. 14. Combine one-mode databases to get single two-mode database. This database will be used here and again in Section 5.1.2.2.

The combined database contains all of the information necessary for calibration except the mode-split targets, which are already known. Preliminary to the iterative fit in the calibration step, a simple model of the Route Planner's rational choice in the base-year must be constructed. The model chosen for the case study is a constant probability, independent of all demographic and other variables.

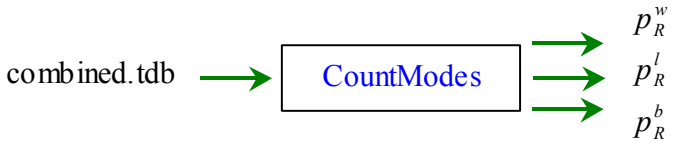

Fig. 15. Calculate Route Planner rational-choice probabilities.

The *CountModes* script calculates the fraction of selected tours in the combined database that have chosen each of the three subsets—walk, bus, or light rail—of the general transit mode when using the general transit mode. These numbers are the simplest estimates of the p_R^k values and are independent of all demographic information.

5.1.2.2 Compute Model Parameter

With the a_c, w_c, b_c, and l_c parameters computed in Section 5.1.1.2, and the p_R^k functions determined in Section 5.1.2.1, the target mode splits can be calibrated by adjusting the

single model parameters α, in each stratum to match the targets computed from Equations 14-16 in Section 2. This is done by first dividing the combined 2-mode tour database, *combined.tdb*, into separate files for each transit mode of each stratum (that is, one file for every calibration parameter). This is shown in Fig. 16.

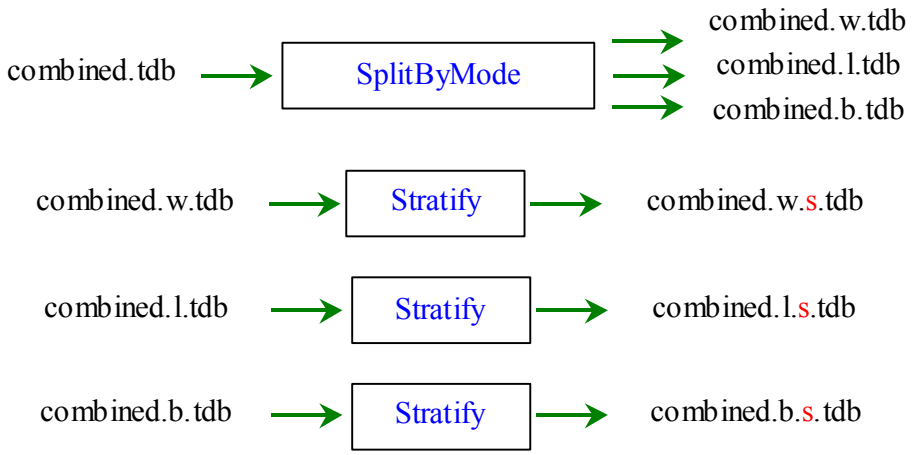

Fig. 16. Split databases by transit choice; stratify each.

The combined database is split into three separate files: one for each of the transit modes the Route Planner may choose when given the general transit mode in the activities. Each of these databases is then divided into 18 separate files, one for each of the strata used in ONDN. The stratum is denoted in the figure by the red *s*, which is a three-character string. The first character is either *w* or *n*, corresponding to work or non-work tours. The second character is a number from one to three corresponding to the home urbanization value (1 = very-urban, 2 = urban, 3 = suburban). The third character is the primary anchor urbanization value. Thus, *s = w11* would be the stratum with only work tours where the home location is in a very urban area and the primary anchor location is in a very urban area.

With the combined database split into the 54 independently calibrated strata, calibration is straightforward. The *TestAlpha* script counts the modes used in each tour resulting from a given value of α, and is used inside an iterative fitting procedure to reach the target. In the Study, a manual fit is used, where the analyst chooses the values of α to be tested and decides when the target mode split has been achieved. A simple bisection method works well. Regardless of the method of fitting, different values of α are tested until the desired mode split is reached.

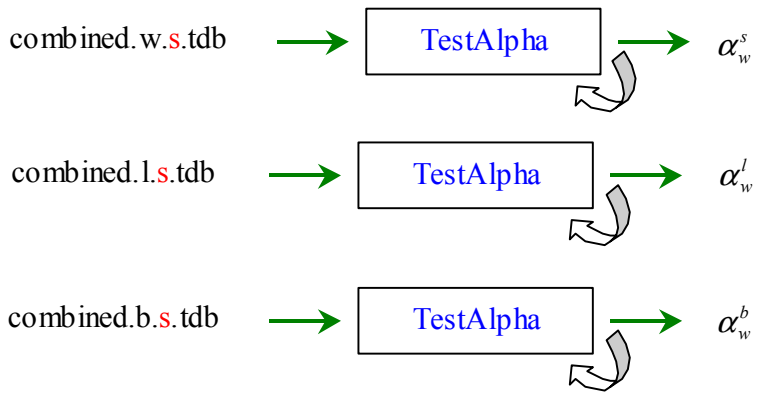

Fig. 17. Fit calibration parameters to data.

A typical relationship between the mode split and the calibration parameter α is shown in the results section. After all strata are fit, there are 18 triples of calibration parameters describing the mode choice model.

5.2 Application to Forecast Year

Application to the forecast year requires that the forecast year activities be moded on transit so that the Route Planner "rational choice" is known. A subset of these tours is routed with an auto mode to determine the mode-choice preference distribution for each stratum using the model parameters calibrated in the base year. The resulting forecast-year preference distributions are used to select a mode for every tour. Afterwards, a similar procedure is used to correct bad choices, while preserving the number of tours using each mode.

5.2.1 Initial Application

The initial application produces a set of forecast-year activities that have modes chosen according to the model calibrated in Section 5.1.

5.2.1.1 Route on Transit

It is assumed that the starting point is a set of forecast-year activities where all walk or rail modes are changed to general transit, as was done for AS7 to create the *A-init* activity set. This initial set of forecast-year activities is called *A-FY*.

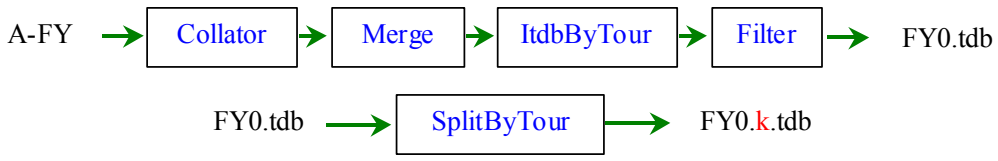

Fig. 18. Create initial forecast year database, then split into files with all first tours in k=1, all second tours in k-3, etc.

As in calibration, all mode feedback scripts operate on tour-based databases. The A-FY activities are used to create the initial forecast-year tour database *FY0.tdb*, as shown in Fig. 18. This is the same set of scripts and programs used to create tour databases from activities used in every earlier instance of this report. Since all tours must be routed on transit mode, there will be multiple tours from a single traveler, and the database must be split into separate databases for each tour. This is again done using the *SplitByTour* script, which creates databases *FY0.k.tdb*. The red *k* in Fig. 18 designates the tour number and can be any integer up to the maximum number of tours for any traveler.

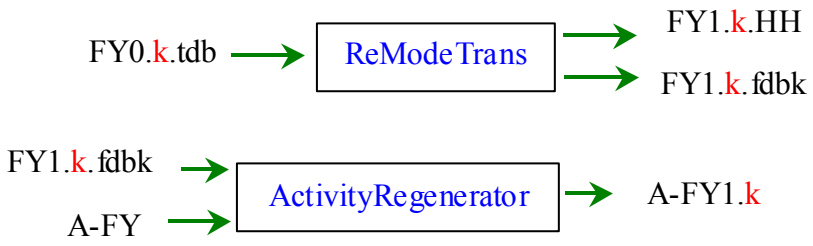

Fig. 19. Convert all selected tours to general transit mode.

The entire initial application stage of mode feedback is done once for each value of *k*, and merged together at the end. Fig. 19 shows the process of changing of activity modes to transit. For each *k*, the tour database is run through the *ReModeTrans* script, which creates a feedback command that re-modes all auto trips to transit for every tour in the database. The Activity Regenerator feedback command file is called *FY1.k.fdbk*, and the corresponding household file is *FY1.k.HH*. That feedback command file is used with the initial forecast-year activities in the Activity Regenerator to create a partial set of activities—all activities for every household in *FY1.k.HH*, with all mode changes in the feedback command file. The partial activity file *A-FY1.k* is not merged into the full activity file because every tour is considered for one value of *k*.

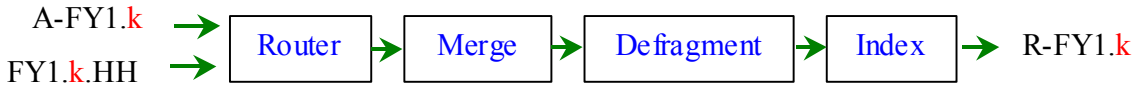

Fig. 20. Route selected tours on general transit mode.

The routing of the transit mode tours is shown in Fig. 20. The procedure is the same as that described in Section 5.1.1.3. The result is a corresponding set of routes labeled *R-FY1.k*.

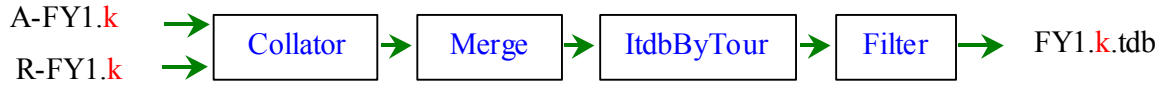

Fig. 21. Create tour database for transit mode tours.

The activities and routes for each *k* are combined into a tour database as in Fig. 22. The result is a set of databases called *FY1.k.tdb*.

Fig. 22. Merge independent-tour databases into single ONDN databases.

The final step in preparing the transit mode data for every tour in ONDN is to merge the databases for every value of *k*. The *FY0.k.tdb* is used as the key that determines from which *FY1.k.tdb* each tour should be drawn. The result is the *FY1.tdb* tour database, where every tour is on transit mode, but the time estimate for each tour was made under the assumption that only that tour (for the given traveler) was re-moded onto transit. That is, for a traveler who had five auto tours, the first tour's transit time is determined under the assumption that the second through fifth tours were still using auto; the second tour's transit time assumes tours 1 and 3-5 were auto, etc.

5.2.1.2 Route Sample on Auto

It is not necessary to test every tour on both auto and transit, only a subset that is sufficiently large for good statistics in estimating modes (Section 5.2.1.3).

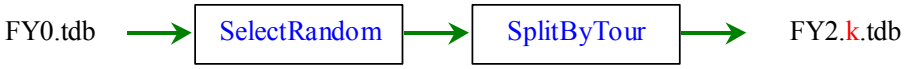

Fig. 23. Select a subset of tours, then split into files with all first tours in k=1, all second tours in k=3, etc.

Before splitting the *FY0.tdb* initial forecast year tour database by tour number, a subset is drawn using *SelectRandom* and a specified fraction, shown in Fig. 23. The chosen fraction is large enough that all strata will have sufficient samples. Note that it would be more efficient to stratify and then sample, thus ensuring good statistics, but the approach described here and in Section 5.1.1.1 is simpler to explain. The resulting databases, split by the tour number, are *FY2.k.tdb*. This set of files is actually a subset of the *FY0.k.tdb* files, and the sample could have been taken from those instead, eliminating the *SplitByTour* step above.

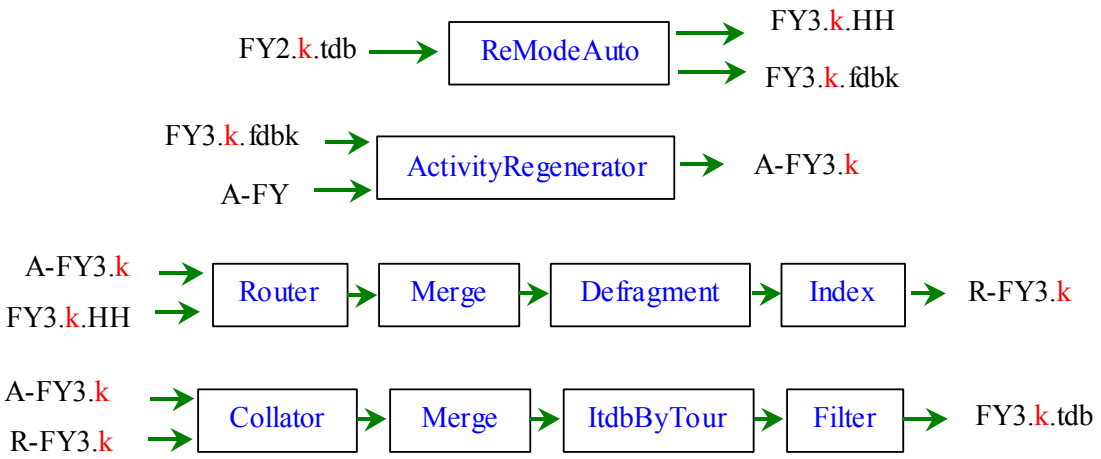

Fig. 24. Route selected tours on auto mode and create tour database.

Fig. 24 illustrates the generation of auto mode data for the mode preference distribution. Since most of the tours will already be on auto mode in the original forecast year activities, this step could be done more efficiently by not re-moding those tours already on auto. However, the staff time necessary to do that and to check that it is working, far outweighs the time it takes to simply re-calculate the auto mode activities. Starting from the tour database split by tour number, *FY2.k.tdb*, feedback commands are created to re-mode every tour in the database onto auto. These are used with the original forecast year activities to create the re-moded activities *A-FY3.k*. The new activities are routed (*R-FY3.k*), and the activities and routes are used to create a new tour database, *FY3.tdb*.

Fig. 25. Merge independent-tour databases into single ONDN databases.

Finally, the auto mode tour databases (*FY3.k.tdb*) are used with the databases that designate the selected activities (*FY2.k.tdb*) to combine the tour number files into a single database having all selected auto mode tours.

5.2.1.3 Compute New Modes

The transit mode and auto mode tour databases are combined to form a single database having information on both auto and transit modes for each of the tours selected for testing. Costs are calculated using the model parameters calibrated in the base year. The distribution of 2-mode costs is used to determine (probabilistically) the mode for each tour using only the transit cost for that tour. The result is a set of feedback commands.

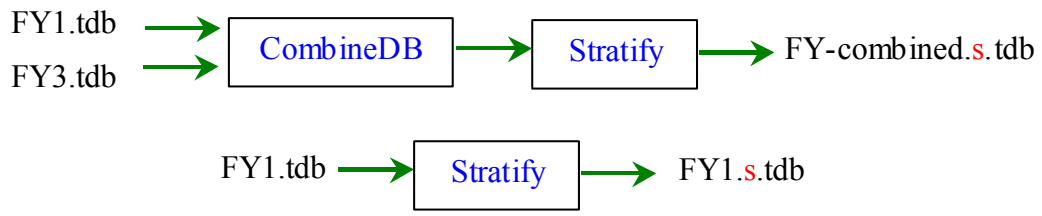

Fig. 26. Combine one-mode databases to get single two-mode database; stratify.

Fig. 26 shows the combination of the two 1-mode databases into a database having information on both modes for each tested tour. The result is stratified according to the method described in Section 5.1.2.2, creating the 18 *FY-combined.s.tdb* files, where the red *s* represents the 18 strata. The transit mode database is also split according to the stratification specification, resulting in the 18 *FY1.s.tdb* files.

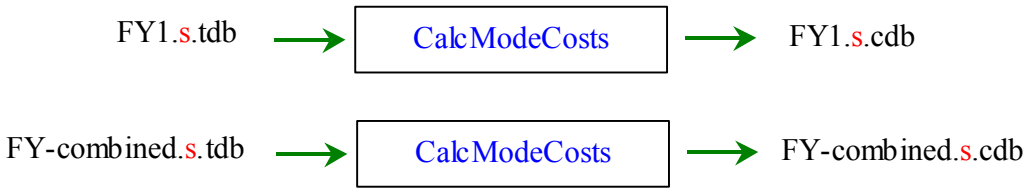

Fig. 27. Calculate tour cost using parameters from Section 5.1.2.2.

For each stratum, the transit mode tour database *FY1.s.tdb* is used with the model parameters for that stratum (s) to create the respective table of tour costs, *FY1.s.cdb*. Since the database is not split according to transit modes (walk, rail, or bus), the *CalcModeCosts* script requires that all three model parameters for that stratum be given as arguments. Then, if the transit mode is walk, the walk α for that stratum is used, etc. The same is done for the combined database *FY-combined.s.tdb*, creating a two-cost table (*FY-combined.s.cdb*) for each stratum.

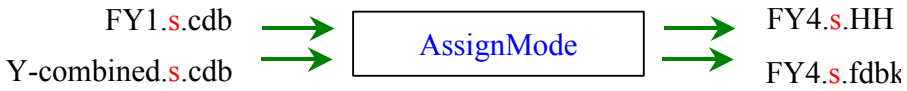

Fig. 28. Assign new modes.

Using the *FY-combined.s.cdb* file as the distribution of costs for each of the three transit modes, the *AssignMode* program examines each tour in the *FY1.s.cdb* database and chooses a new mode. If the transit mode is walk, then the distribution for walk will be used, etc. Output is a feedback command for every tour in the transit database specifying a mode and a corresponding household file.

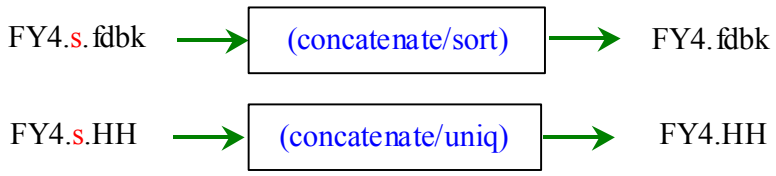

Fig. 29. Combine feedback files.

Since each tour is in only one stratum, the tours in the 18 feedback files are independent. There is no header in these files, so the combined file *FY4.fdbk* is created by concatenating all of the files and sorting by traveler and tour. The household file must also be "uniqued" after concatenation so that each household appears only once. The result is the complete set of feedback commands for the initial run of mode feedback.

5.2.1.4 Route Chosen Modes

The modes computed in the previous section are used in the Activity Regenerator with the original forecast-year activities to create the initial set of activities with modes determined according to the designated mode-choice model. As shown in Fig. 30, the resulting activity file is labeled *A-FY4*. This file has all activities for every household included in the feedback file.

Fig. 30. Convert all selected tours to general transit mode.

The final step to application of the mode choice model is routing the tours on the selected modes, as shown in Fig. 31.

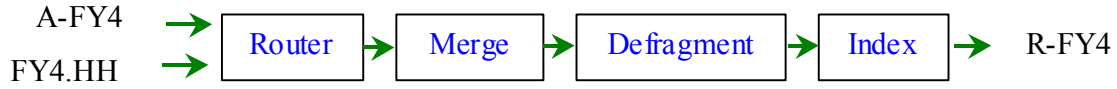

Fig. 31. Route all ONDN tours on chosen mode.

The router household file contains entries for every household in ONDN. The resulting route file includes all routes for ONDN. If no iteration is necessary, then *R-FY4* are the routes for the final set of mode choices for every tour in ONDN.

5.2.2 Iteration

It is unlikely that there are no problems in *R-FY4*, so iteration of the mode choice methodology is required. The first step is identifying all of the problems, and the second is fixing them.

5.2.2.1 Initial Evaluation & Correction

Evaluation is a simple process. The cost for every tour is calculated, and the number of transfers in every transit tour is calculated. Any rule could be used to declare a tour infeasible. In the Study, an infeasible transit tour is any transit tour lasting longer than 24 hours or having more than three transit transfers on any trip within the tour. An infeasible auto trip is any auto tour lasting more than 24 hours.

The first step in the process is the creation of the tour database. The information on infeasible mode selections comes from the actual mode choice in *R-FY4*. The creation of the tour database *FY4.tdb* is shown in Fig. 32.

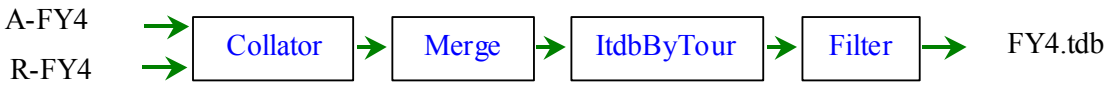

Fig. 32. Create tour database for all ONDN tours.

However, when a tour is switched from auto to transit to account for the infeasible transit tours, only those tours that will result in feasible transit tours should be considered as choices. This information comes from the tour database for *R-FY1* (already calculated in *FY1.tdb*), which includes the transit information for all tours. All of the information necessary for correcting bad mode choices is combined into a single database using the *CombineDB* script, as shown in Fig. 33. So that the correction program does not have to know how to group the tours, the database is stratified and split by transit mode, creating the 3x18 *Fix1.s.m.tdb* databases, where *s* represents the three-character group identifier and *m* designates the mode used when on transit (walk, rail, or bus).

Fig. 33. Combine one-mode databases to get two-mode databases; stratify.

Since tours are considered independently in FY1 but may end up together in FY4 or in a fixed set of modes, the FY1 database is not a good measure of a tour's unfeasibility in this case. For the Study , an assumption is made that these compositions do not result in problems that cannot be corrected through the random sampling of the correction process. That is, their numbers are few.

With each stratum considered independently, the mode split is preserved within a stratum as well as overall. The combined database contains information about the mode chosen and information about transit modes. However, if a tour is by transit, there is no information about auto mode for that tour. It is assumed that there will be few infeasible auto tours. In the unlikely event that one does occur, almost any other tour could be switched to auto to correct for it.

Fig. 34. Calculate tour costs using parameters from Section 5.1.2.2.

Costs are calculated from the combined tour database using the *CalcModeCosts* script. The resulting cost databases (*Fix1.s.cdb*) are used in the program that makes the corrections while preserving mode split. Each infeasible tour is re-moded. If it was an auto tour, then it becomes general transit, and if transit then auto. All changes are counted, and the same number of tours is changed to the other mode, so that the net count of tour modes remains unchanged. The output is a set of feedback commands to implement these changes.

Fig. 35. Fix infeasible modes.

The feedback files are combined, so that the Activity Regenerator and the Route Planner may apply all changes to every household at once. This is the same as was done to create FY4, and is shown in Fig. 36.

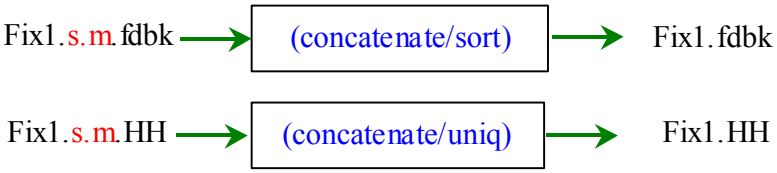

Fig. 36. Combine feedback files.

The final step to correction is running the Activity Generator and Route Planner to create a new set of forecast year activities and routes. There is now an additional merging step. The changes specified in the "fix" feedback commands should be much fewer than the total number of tours. It is, therefore, more efficient to regenerate and route only those households having changes, and then merge them together with the unchanged households. This is done using the *MergeIndices* tool for the activities and *PlanFilter* for the routes. The merged indexes are then defragmented and re-indexed. The result is a single complete set of activities and routes for all households having any ONDN tours, *A-Fix1* and *R-Fix1*.

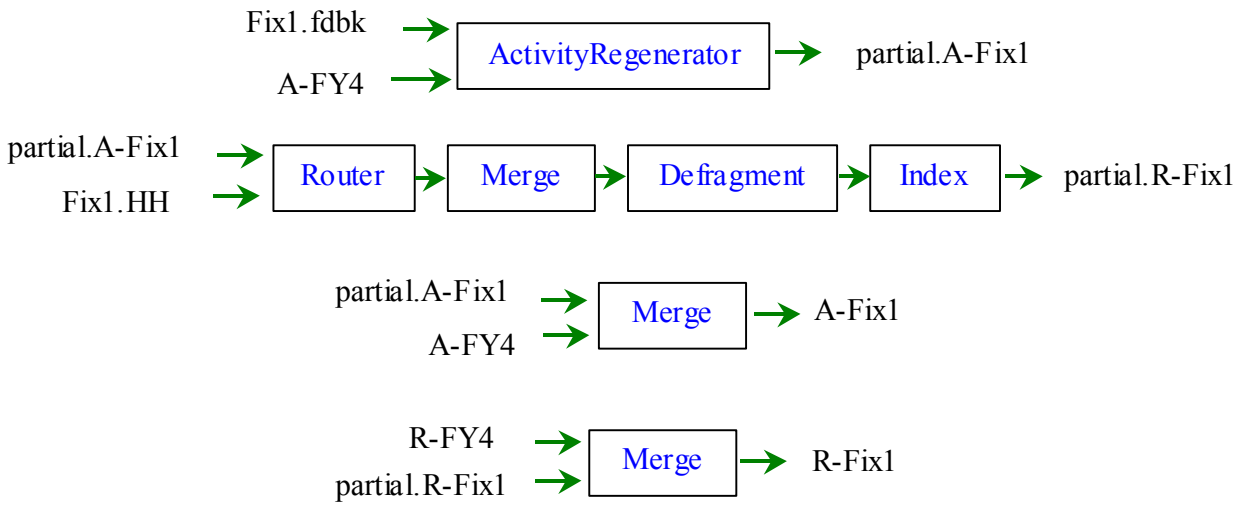

Fig. 37. Regenerated and route all corrected tours.

If the routes in *R-Fix1* have no infeasible tours, they can be merged with those produced by routing all household not having ONDN tours to produce the complete case study set of routes.

5.2.2.2 Continued Correction

The process described in Section 5.2.2.1 may be repeated indefinitely by simply changing the file names. Fig. 38 shows the procedure for the next iteration.

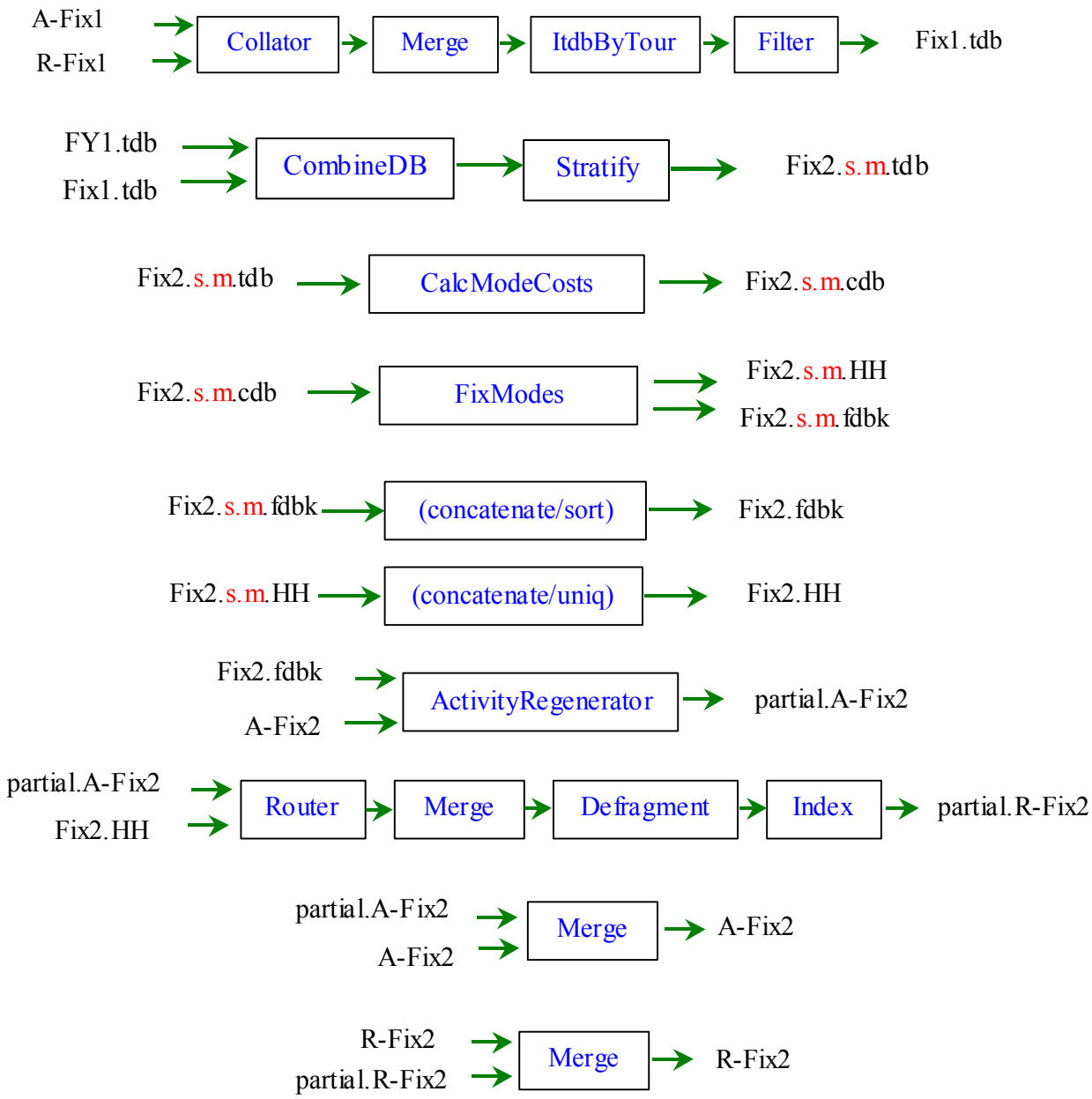

Fig. 38. The second correction iteration.

6. Appendix B: Scripts and Configuration Files

6.1 Configuration Files

6.1.1 ReModeAuto-A.cfg

```
TRANSIMS_ROOT                     /home/transims/CaseStudy3/scenarios/allstr

CONFIG_DEFAULT_FILE               $TRANSIMS_ROOT/allstr.cfg

############################################################
# recent corrections:

ACT_TRAVEL_TIMES_FILE             $TRANSIMS_ROOT/activity/traveltime_050101
ACT_TRAVEL_TIME_FUNCTION_MODES    1;2;3;4;5;6;7

ACT_SURVEY_ACTIVITY_FILE          $TRANSIMS_ROOT/data/survey_activities
ACT_SURVEY_WEIGHTS_FILE           $TRANSIMS_ROOT/data/survey_weights_25
#ACT_TRAVEL_TIME_INTERVALS_FILE   $TRANSIMS_ROOT/activity/time_intervals

LOG ACT                           1

############################################################
# configuration file keys for ONDN activity feedback run:

ACTIVITY_FILE      $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.act
ACT_FEEDBACK_FILE  $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/auto.fdbk

ACT_PARTIAL_OUTPUT $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/auto.act
ACT_PROBLEM_FILE   $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/auto.ActivityRegenerator.problems
ACT_LOG_FILE       $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/auto.ActivityRegenerator.log

############################################################
# rockhopper changes:

# 24 hours
ROUTER_MAX_LEG_LENGTH           86400

# used for park and ride
ROUTER_MAX_NODES_EXAMINED       5000000

NET_ACTIVITY_LOCATION_TABLE     Activity_Location.20010806.tbl
NET_DETECTOR_TABLE              Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                  Link.20011119.tbl
NET_NODE_TABLE                  Node.20010806.tbl
NET_PARKING_TABLE               Parking.20010806.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE          Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE           $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
```

6.1.2 ReModeTrans-A.cfg

```
TRANSIMS_ROOT                   /home/transims/CaseStudy3/scenarios/allstr

CONFIG_DEFAULT_FILE             $TRANSIMS_ROOT/allstr.cfg

############################################################
# recent corrections:

ACT_TRAVEL_TIMES_FILE            $TRANSIMS_ROOT/activity/traveltime_050101
ACT_TRAVEL_TIME_FUNCTION_MODES   1;2;3;4;5;6;7

ACT_SURVEY_ACTIVITY_FILE         $TRANSIMS_ROOT/data/survey_activities
ACT_SURVEY_WEIGHTS_FILE          $TRANSIMS_ROOT/data/survey_weights_25
#ACT_TRAVEL_TIME_INTERVALS_FILE  $TRANSIMS_ROOT/activity/time_intervals

LOG_ACT                          1

############################################################
# configuration file keys for ONDN activity feedback run:

ACTIVITY_FILE        $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/no-subtours.act
ACT_FEEDBACK_FILE    $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.fdbk

ACT_PARTIAL_OUTPUT   $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.act
ACT_PROBLEM_FILE     $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.ActivityRegenerator.problems
ACT_LOG_FILE         $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.ActivityRegenerator.log

############################################################
# rockhopper changes:

# 24 hours
ROUTER_MAX_LEG_LENGTH           86400

# used for park and ride
ROUTER_MAX_NODES_EXAMINED       5000000

NET_ACTIVITY_LOCATION_TABLE     Activity_Location.20010806.tbl
NET_DETECTOR_TABLE              Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                  Link.20011119.tbl
NET_NODE_TABLE                  Node.20010806.tbl
NET_PARKING_TABLE               Parking.20010806.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE          Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE           $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
```

6.1.3 RemoveSubtours-A.cfg

```
TRANSIMS_ROOT                     /home/transims/CaseStudy3/scenarios/allstr

CONFIG_DEFAULT_FILE               $TRANSIMS_ROOT/allstr.cfg

############################################################
# recent corrections:

ACT_TRAVEL_TIMES_FILE             $TRANSIMS_ROOT/activity/traveltime_050101
ACT_TRAVEL_TIME_FUNCTION_MODES    1;2;3;4;5;6;7

ACT_SURVEY_ACTIVITY_FILE          $TRANSIMS_ROOT/data/survey_activities
ACT_SURVEY_WEIGHTS_FILE           $TRANSIMS_ROOT/data/survey_weights_25
#ACT_TRAVEL_TIME_INTERVALS_FILE   $TRANSIMS_ROOT/activity/time_intervals

LOG_ACT                           1

############################################################
# configuration file keys for ONDN activity feedback run:

ACTIVITY_FILE        $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/sample4.act
ACT_FEEDBACK_FILE    $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/no-subtours.fdbk

ACT_PARTIAL_OUTPUT   $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/no-subtours.partial-act
ACT_PROBLEM_FILE     $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/no-subtours.ActivityRegenerator.problems
ACT_LOG_FILE         $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/no-subtours.ActivityRegenerator.log

############################################################
# rockhopper changes:

# 24 hours
ROUTER_MAX_LEG_LENGTH           86400

# used for park and ride
ROUTER_MAX_NODES_EXAMINED       5000000

NET_ACTIVITY_LOCATION_TABLE     Activity_Location.20010806.tbl
NET_DETECTOR_TABLE              Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                  Link.20011119.tbl
NET_NODE_TABLE                  Node.20010806.tbl
NET_PARKING_TABLE               Parking.20010806.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE          Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE           $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

###################################
```

6.1.4 final2-A.cfg

```
TRANSIMS_ROOT                    /home/transims/CaseStudy3/scenarios/allstr

CONFIG_DEFAULT_FILE              $TRANSIMS_ROOT/allstr.cfg

############################################################
# recent corrections:

ACT_TRAVEL_TIMES_FILE            $TRANSIMS_ROOT/activity/traveltime_050101
ACT_TRAVEL_TIME_FUNCTION_MODES   1;2;3;4;5;6;7

ACT_SURVEY_ACTIVITY_FILE         $TRANSIMS_ROOT/data/survey_activities
ACT_SURVEY_WEIGHTS_FILE          $TRANSIMS_ROOT/data/survey_weights_25
#ACT_TRAVEL_TIME_INTERVALS_FILE  $TRANSIMS_ROOT/activity/time_intervals

LOG_ACT                          1

############################################################
# configuration file keys for ONDN activity feedback run:

ACTIVITY_FILE       $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/application/auto.act
ACT_FEEDBACK_FILE   $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final2.fdbk

ACT_PARTIAL_OUTPUT  $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final2.partial-act
ACT_PROBLEM_FILE    $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final2.ActivityRegenerator.problems
ACT_LOG_FILE        $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final2.ActivityRegenerator.log

############################################################
# rockhopper changes:

# 24 hours
ROUTER_MAX_LEG_LENGTH           86400

# used for park and ride
ROUTER_MAX_NODES_EXAMINED       5000000

NET_ACTIVITY_LOCATION_TABLE     Activity_Location.20010806.tbl
NET_DETECTOR_TABLE              Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                  Link.20011119.tbl
NET_NODE_TABLE                  Node.20010806.tbl
NET_PARKING_TABLE               Parking.20010806.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE          Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE           $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
```

6.1.5 final1-R.cfg

```
TRANSIMS_ROOT                    /home/transims/CaseStudy3/scenarios/allstr

CONFIG_DEFAULT_FILE              $TRANSIMS_ROOT/allstr.cfg

############################################################
# keys for ONDN re-route run:

ACTIVITY_FILE                    $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final1.act
#ROUTER_LINK_DELAY_FILE          $TRANSIMS_ROOT/feedback/mode/ONDN/summary.tim

ROUTER_PROBLEM_FILE              $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final1.Router.problems
PLAN_FILE                        $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final1.plans
ROUTER_COMPLETED_HOUSEHOLD_FILE  $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final1.completed-HH

ROUTER_DELAY_NOISE               0.10
ROUTER_OVERDO                    0.0
ROUTER_SEED                      912345678

############################################################
# configuration file keys for better performance & results:

ROUTER_DISPLAY_PATHS          0         # excessive output = 1
ROUTER_INTERNAL_PLAN_SIZE     2000      # default = 400
ROUTER_MAX_NODES_EXAMINED     5000000   # default = 400000
ROUTER_MESSAGE_LEVEL          1         # severe=0, everything=1
ROUTER_NUMBER_THREADS         2         # per box
LOG_ROUTING                   1         # for speed calc
ROUTER_MAX_LEG_LENGTH         86400     # max 24 hour legs = basically unlimited
# This should limit route hopping without impacting route generation:
ROUTER_GET_ON_TRANSIT_DELAY   240
ROUTER_GET_OFF_TRANSIT_DELAY  240

############################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE     Activity_Location.20010806.tbl
NET_DETECTOR_TABLE              Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                  Link.20011119.tbl
NET_NODE_TABLE                  Node.20010806.tbl
NET_PARKING_TABLE               Parking.20010806.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE          Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE           $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
# new configuration file key from Paula:

ROUTER_MAX_TRIP_TIME            172800
```

6.1.6 final0-R.cfg

```
TRANSIMS_ROOT                    /home/transims/CaseStudy3/scenarios/allstr

CONFIG_DEFAULT_FILE              $TRANSIMS_ROOT/allstr.cfg

############################################################
# configuration file keys for ONDN re-route run:

ACTIVITY_FILE                    $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final0.act
#ROUTER_LINK_DELAY_FILE          $TRANSIMS_ROOT/feedback/mode/ONDN/summary.tim

ROUTER_PROBLEM_FILE              $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final0.Router.problems
PLAN_FILE                        $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final0.plans
ROUTER_COMPLETED_HOUSEHOLD_FILE  $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final0.completed-HH

ROUTER_DELAY_NOISE               0.10
ROUTER_OVERDO                    0.0
ROUTER_SEED                      912345678

############################################################
# configuration file keys for better performance & results:

ROUTER_DISPLAY_PATHS          0        # excessive output = 1
ROUTER_INTERNAL_PLAN_SIZE     2000     # default = 400
ROUTER_MAX_NODES_EXAMINED     5000000  # default = 400000
ROUTER_MESSAGE_LEVEL          1        # severe=0, everything=1
ROUTER_NUMBER_THREADS         2        # per box
LOG_ROUTING                   1        # for speed calc
ROUTER_MAX_LEG_LENGTH         86400    # max 24 hour legs = basically unlimited
# This should limit route hopping without impacting route generation:
ROUTER_GET_ON_TRANSIT_DELAY   240
ROUTER_GET_OFF_TRANSIT_DELAY  240

############################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE     Activity_Location.20010806.tbl
NET_DETECTOR_TABLE              Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                  Link.20011119.tbl
NET_NODE_TABLE                  Node.20010806.tbl
NET_PARKING_TABLE               Parking.20010806.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE          Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE           $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
# new configuration file key from Paula:

ROUTER_MAX_TRIP_TIME            172800
```

6.1.7 test-C.cfg

```
########################################################################

# This file has Collator keys for creating an iteration database to be
# converted by ItdbByTour.pl into a "by-tour" database.  -JPS

########################################################################

TRANSIMS_ROOT                   /home/transims/CaseStudy3/scenarios/allstr/
#CONFIG_DEFAULT_FILE            $TRANSIMS_ROOT/allstr.cfg

NET_DIRECTORY                   $TRANSIMS_ROOT/network
NET_NODE_TABLE                  Node.tbl
NET_LINK_TABLE                  Link.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.tbl
NET_PARKING_TABLE               Parking.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.tbl
NET_TIMING_PLAN_TABLE           Timing_Plan.tbl
NET_SPEED_TABLE                 Speed.tbl
NET_LANE_USE_TABLE              Lane_Use.tbl
NET_TRANSIT_STOP_TABLE          Transit_Stop.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.tbl
NET_DETECTOR_TABLE              Detector.tbl
NET_TURN_PROHIBITION_TABLE      Turn_Prohibition.tbl
NET_BARRIER_TABLE               Barrier.tbl
NET_ACTIVITY_LOCATION_TABLE     Activity_Location.tbl
NET_PROCESS_LINK_TABLE          Process_Link.tbl

ACT_HOME_ACTIVITY_TYPE          0
ACT_WORK_ACTIVITY_TYPE          1
ACT_SCHOOL_ACTIVITY_TYPE        7

ACT_ANCHOR_ACTIVITY_TYPE_1      0   # Home
ACT_ANCHOR_ACTIVITY_TYPE_2      1   # Work
ACT_ANCHOR_ACTIVITY_TYPE_3      7   # School
ACT_ANCHOR_ACTIVITY_TYPE_4      8   # College

############################################################
# configuration file keys for parallel runs:

#ACT_POPULATION_FILE      $TRANSIMS_ROOT/population/pop_converted
ACT_POPULATION_FILE       $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/sample4.pop

TRANSIT_ROUTE_FILE        $TRANSIMS_ROOT/network/Transit_Route.tbl

############################################################

#SEL_POPULATION_FILE      $TRANSIMS_ROOT/population/pop_located
SEL_POPULATION_FILE       $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/sample4.pop
VEHICLE_FILE              $TRANSIMS_ROOT/vehicle/vehicles.all
SEL_PLAN_FILE             $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/test.plans
SEL_ACTIVITY_FILE         $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.act
#SEL_EVENT_FILE           $TRANSIMS_ROOT/output/anomaly.offplan.trv

SEL_ITDB_FILE             $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/test

SEL_MESSAGE_LEVEL         0
SEL_NO_ITDB_INDEX         1

############################################################
# Data gathered by the Collator for feedback

# Pop data
SEL_USE_RHHINC                  1
SEL_USE_AGE                     1
```

```
# Act data
SEL_USE_START_ACT_LOCATION       1
SEL_USE_END_ACT_TYPE             1
SEL_USE_END_MODE_PREF            1
SEL_USE_END_ACT_LOCATION         1
SEL_USE_END_OTHER_PARTICIPANTS   1
SEL_USE_END_DUR_UB               1
SEL_USE_END_DUR_LB               1
SEL_USE_DRIVER_ID                1

# Activity Location Table data
SEL_USE_START_ACT_USER_DATA      Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8
SEL_USE_END_ACT_USER_DATA        Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8

# Route data
SEL_USE_EUCLID           1
SEL_USE_DURATION         1
SEL_USE_MODE_STRING      1

# event data
#SEL_USE_OFF_PLAN         1

################################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE     Activity_Location.20010806.tbl
NET_DETECTOR_TABLE              Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                  Link.20011119.tbl
NET_NODE_TABLE                  Node.20010806.tbl
NET_PARKING_TABLE               Parking.20010806.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE          Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE           $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

################################################################
```

6.1.8 transit-C.cfg

```
########################################################################

# This file has Collator keys for creating an iteration database to be
# converted by ItdbByTour.pl into a "by-tour" database.  -JPS

########################################################################

TRANSIMS_ROOT                    /home/transims/CaseStudy3/scenarios/allstr/
#CONFIG_DEFAULT_FILE             $TRANSIMS_ROOT/allstr.cfg

NET_DIRECTORY                    $TRANSIMS_ROOT/network
NET_NODE_TABLE                   Node.tbl
NET_LINK_TABLE                   Link.tbl
NET_POCKET_LANE_TABLE            Pocket_Lane.tbl
NET_PARKING_TABLE                Parking.tbl
NET_LANE_CONNECTIVITY_TABLE      Lane_Connectivity.tbl
NET_UNSIGNALIZED_NODE_TABLE      Unsignalized_Node.tbl
NET_SIGNALIZED_NODE_TABLE        Signalized_Node.tbl
NET_PHASING_PLAN_TABLE           Phasing_Plan.tbl
NET_TIMING_PLAN_TABLE            Timing_Plan.tbl
NET_SPEED_TABLE                  Speed.tbl
NET_LANE_USE_TABLE               Lane_Use.tbl
NET_TRANSIT_STOP_TABLE           Transit_Stop.tbl
NET_SIGNAL_COORDINATOR_TABLE     Signal_Coordinator.tbl
NET_DETECTOR_TABLE               Detector.tbl
NET_TURN_PROHIBITION_TABLE       Turn_Prohibition.tbl
NET_BARRIER_TABLE                Barrier.tbl
NET_ACTIVITY_LOCATION_TABLE      Activity_Location.tbl
NET_PROCESS_LINK_TABLE           Process_Link.tbl

ACT_HOME_ACTIVITY_TYPE           0
ACT_WORK_ACTIVITY_TYPE           1
ACT_SCHOOL_ACTIVITY_TYPE         7

ACT_ANCHOR_ACTIVITY_TYPE_1       0 #     Home
ACT_ANCHOR_ACTIVITY_TYPE_2       1 #     Work
ACT_ANCHOR_ACTIVITY_TYPE_3       7 #     School
ACT_ANCHOR_ACTIVITY_TYPE_4       8 #     College

############################################################
# configuration file keys for parallel runs:

#ACT_POPULATION_FILE     $TRANSIMS_ROOT/population/pop_converted
ACT_POPULATION_FILE      $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/sample4.pop

TRANSIT_ROUTE_FILE       $TRANSIMS_ROOT/network/Transit_Route.tbl

############################################################

#SEL_POPULATION_FILE    $TRANSIMS_ROOT/population/pop_located
SEL_POPULATION_FILE     $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/sample4.pop
VEHICLE_FILE            $TRANSIMS_ROOT/vehicle/vehicles.all
SEL_PLAN_FILE           $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.plans
SEL_ACTIVITY_FILE       $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.act
#SEL_EVENT_FILE         $TRANSIMS_ROOT/output/anomaly.offplan.trv

SEL_ITDB_FILE           $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit

SEL_MESSAGE_LEVEL       0
SEL_NO_ITDB_INDEX       1

############################################################
# Data gathered by the Collator for feedback

# Pop data
SEL_USE_RHHINC          1
SEL_USE_AGE             1
```

```
# Act data
SEL_USE_START_ACT_LOCATION        1
SEL_USE_END_ACT_TYPE              1
SEL_USE_END_MODE_PREF             1
SEL_USE_END_ACT_LOCATION          1
SEL_USE_END_OTHER_PARTICIPANTS    1
SEL_USE_END_DUR_UB                1
SEL_USE_END_DUR_LB                1
SEL_USE_DRIVER_ID                 1

# Activity Location Table data
SEL_USE_START_ACT_USER_DATA     Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8
SEL_USE_END_ACT_USER_DATA       Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8

# Route data
SEL_USE_EUCLID        1
SEL_USE_DURATION      1
SEL_USE_MODE_STRING   1

# event data
#SEL_USE_OFF_PLAN     1

################################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE     Activity_Location.20010806.tbl
NET_DETECTOR_TABLE              Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                  Link.20011119.tbl
NET_NODE_TABLE                  Node.20010806.tbl
NET_PARKING_TABLE               Parking.20010806.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE          Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE           $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

################################################################
```

6.1.9 final2-C.cfg

```
########################################################################

# This file has Collator keys for creating an iteration database to be
# converted by ItdbByTour.pl into a "by-tour" database.  -JPS

########################################################################

TRANSIMS_ROOT                   /home/transims/CaseStudy3/scenarios/allstr/
#CONFIG_DEFAULT_FILE            $TRANSIMS_ROOT/allstr.cfg

NET_DIRECTORY                   $TRANSIMS_ROOT/network
NET_NODE_TABLE                  Node.tbl
NET_LINK_TABLE                  Link.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.tbl
NET_PARKING_TABLE               Parking.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.tbl
NET_TIMING_PLAN_TABLE           Timing_Plan.tbl
NET_SPEED_TABLE                 Speed.tbl
NET_LANE_USE_TABLE              Lane_Use.tbl
NET_TRANSIT_STOP_TABLE          Transit_Stop.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.tbl
NET_DETECTOR_TABLE              Detector.tbl
NET_TURN_PROHIBITION_TABLE      Turn_Prohibition.tbl
NET_BARRIER_TABLE               Barrier.tbl
NET_ACTIVITY_LOCATION_TABLE     Activity_Location.tbl
NET_PROCESS_LINK_TABLE          Process_Link.tbl

ACT_HOME_ACTIVITY_TYPE          0
ACT_WORK_ACTIVITY_TYPE          1
ACT_SCHOOL_ACTIVITY_TYPE        7

ACT_ANCHOR_ACTIVITY_TYPE_1      0     # Home
ACT_ANCHOR_ACTIVITY_TYPE_2      1     # Work
ACT_ANCHOR_ACTIVITY_TYPE_3      7     # School
ACT_ANCHOR_ACTIVITY_TYPE_4      8     # College

#################################################################
# configuration file keys for parallel runs:

#ACT_POPULATION_FILE    $TRANSIMS_ROOT/population/pop_converted
ACT_POPULATION_FILE     $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/application/sample4.pop

TRANSIT_ROUTE_FILE      $TRANSIMS_ROOT/network/Transit_Route.tbl

#################################################################

#SEL_POPULATION_FILE    $TRANSIMS_ROOT/population/pop_located
SEL_POPULATION_FILE     $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/application/sample4.pop
VEHICLE_FILE            $TRANSIMS_ROOT/vehicle/vehicles.all
SEL_PLAN_FILE           $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final2.plans
SEL_ACTIVITY_FILE       $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final2.act
#SEL_EVENT_FILE         $TRANSIMS_ROOT/output/anomaly.offplan.trv

SEL_ITDB_FILE           $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final2

SEL_MESSAGE_LEVEL       0
SEL_NO_ITDB_INDEX       1

#################################################################
# Data gathered by the Collator for feedback
```

```
# Pop data
SEL_USE_RHHINC          1
SEL_USE_AGE             1

# Act data
SEL_USE_START_ACT_LOCATION        1
SEL_USE_END_ACT_TYPE              1
SEL_USE_END_MODE_PREF             1
SEL_USE_END_ACT_LOCATION          1
SEL_USE_END_OTHER_PARTICIPANTS    1
SEL_USE_END_DUR_UB                1
SEL_USE_END_DUR_LB                1
SEL_USE_DRIVER_ID                 1

# Activity Location Table data
SEL_USE_START_ACT_USER_DATA       Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8
SEL_USE_END_ACT_USER_DATA         Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8

# Route data
SEL_USE_EUCLID          1
SEL_USE_DURATION        1
SEL_USE_MODE_STRING     1

# event data
#SEL_USE_OFF_PLAN       1

############################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE     Activity_Location.20010806.tbl
NET_DETECTOR_TABLE              Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                  Link.20011119.tbl
NET_NODE_TABLE                  Node.20010806.tbl
NET_PARKING_TABLE               Parking.20010806.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE          Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE           $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
```

6.1.10 final0-C.cfg

```
########################################################################

# This file has Collator keys for creating an iteration database to be
# converted by ItdbByTour.pl into a "by-tour" database.  -JPS

########################################################################

TRANSIMS_ROOT                   /home/transims/CaseStudy3/scenarios/allstr/
#CONFIG_DEFAULT_FILE            $TRANSIMS_ROOT/allstr.cfg

NET_DIRECTORY                   $TRANSIMS_ROOT/network
NET_NODE_TABLE                  Node.tbl
NET_LINK_TABLE                  Link.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.tbl
NET_PARKING_TABLE               Parking.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.tbl
NET_TIMING_PLAN_TABLE           Timing_Plan.tbl
NET_SPEED_TABLE                 Speed.tbl
NET_LANE_USE_TABLE              Lane_Use.tbl
NET_TRANSIT_STOP_TABLE          Transit_Stop.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.tbl
NET_DETECTOR_TABLE              Detector.tbl
NET_TURN_PROHIBITION_TABLE      Turn_Prohibition.tbl
NET_BARRIER_TABLE               Barrier.tbl
NET_ACTIVITY_LOCATION_TABLE     Activity_Location.tbl
NET_PROCESS_LINK_TABLE          Process_Link.tbl

ACT_HOME_ACTIVITY_TYPE          0
ACT_WORK_ACTIVITY_TYPE          1
ACT_SCHOOL_ACTIVITY_TYPE        7

ACT_ANCHOR_ACTIVITY_TYPE_1      0     # Home
ACT_ANCHOR_ACTIVITY_TYPE_2      1     # Work
ACT_ANCHOR_ACTIVITY_TYPE_3      7     # School
ACT_ANCHOR_ACTIVITY_TYPE_4      8     # College

#################################################################
# configuration file keys for parallel runs:

#ACT_POPULATION_FILE       $TRANSIMS_ROOT/population/pop_converted
ACT_POPULATION_FILE        $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/application/sample4.pop

TRANSIT_ROUTE_FILE         $TRANSIMS_ROOT/network/Transit_Route.tbl

#################################################################

#SEL_POPULATION_FILE       $TRANSIMS_ROOT/population/pop_located
SEL_POPULATION_FILE        $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/application/sample4.pop
VEHICLE_FILE               $TRANSIMS_ROOT/vehicle/vehicles.all
SEL_PLAN_FILE              $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final0.plans
SEL_ACTIVITY_FILE          $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final0.act
#SEL_EVENT_FILE            $TRANSIMS_ROOT/output/anomaly.offplan.trv

SEL_ITDB_FILE              $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final0

SEL_MESSAGE_LEVEL          0
SEL_NO_ITDB_INDEX          1

#################################################################
# Data gathered by the Collator for feedback
```

```
# Pop data
SEL_USE_RHHINC     1
SEL_USE_AGE        1

# Act data
SEL_USE_START_ACT_LOCATION        1
SEL_USE_END_ACT_TYPE              1
SEL_USE_END_MODE_PREF             1
SEL_USE_END_ACT_LOCATION          1
SEL_USE_END_OTHER_PARTICIPANTS    1
SEL_USE_END_DUR_UB                1
SEL_USE_END_DUR_LB                1
SEL_USE_DRIVER_ID                 1

# Activity Location Table data
SEL_USE_START_ACT_USER_DATA       Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8
SEL_USE_END_ACT_USER_DATA         Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8

# Route data
SEL_USE_EUCLID         1
SEL_USE_DURATION       1
SEL_USE_MODE_STRING    1

# event data
#SEL_USE_OFF_PLAN      1

############################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE        Activity_Location.20010806.tbl
NET_DETECTOR_TABLE                 Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE        Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                     Link.20011119.tbl
NET_NODE_TABLE                     Node.20010806.tbl
NET_PARKING_TABLE                  Parking.20010806.tbl
NET_PHASING_PLAN_TABLE             Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE              Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE             Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE       Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE          Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE        Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE              $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################
```

6.1.11 final1-A.cfg

```
TRANSIMS_ROOT                     /home/transims/CaseStudy3/scenarios/allstr

CONFIG_DEFAULT_FILE               $TRANSIMS_ROOT/allstr.cfg

############################################################
# recent corrections:

ACT_TRAVEL_TIMES_FILE             $TRANSIMS_ROOT/activity/traveltime_050101
ACT_TRAVEL_TIME_FUNCTION_MODES    1;2;3;4;5;6;7

ACT_SURVEY_ACTIVITY_FILE          $TRANSIMS_ROOT/data/survey_activities
ACT_SURVEY_WEIGHTS_FILE           $TRANSIMS_ROOT/data/survey_weights_25
#ACT_TRAVEL_TIME_INTERVALS_FILE   $TRANSIMS_ROOT/activity/time_intervals

LOG_ACT                           1

############################################################
# configuration file keys for ONDN activity feedback run:

ACTIVITY_FILE        $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/application/transit.act
ACT_FEEDBACK_FILE    $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final1.fdbk

ACT_PARTIAL_OUTPUT   $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final1.partial-act
ACT_PROBLEM_FILE     $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final1.ActivityRegenerator.problems
ACT_LOG_FILE         $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final1.ActivityRegenerator.log

############################################################
# rockhopper changes:

# 24 hours
ROUTER_MAX_LEG_LENGTH           86400

# used for park and ride
ROUTER_MAX_NODES_EXAMINED       5000000

NET_ACTIVITY_LOCATION_TABLE     Activity_Location.20010806.tbl
NET_DETECTOR_TABLE              Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                  Link.20011119.tbl
NET_NODE_TABLE                  Node.20010806.tbl
NET_PARKING_TABLE               Parking.20010806.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE          Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE           $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
```

6.1.12 auto-R.cfg

```
TRANSIMS_ROOT               /home/transims/CaseStudy3/scenarios/allstr

CONFIG_DEFAULT_FILE         $TRANSIMS_ROOT/allstr.cfg

############################################################
# keys for ONDN re-route run:

ACTIVITY_FILE                     $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/auto.act
#ROUTER_LINK_DELAY_FILE           $TRANSIMS_ROOT/feedback/mode/ONDN/summary.tim

ROUTER_PROBLEM_FILE               $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/auto.Router.problems
PLAN_FILE                         $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/auto.plans
ROUTER_COMPLETED_HOUSEHOLD_FILE   $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/auto.completed-HH

ROUTER_DELAY_NOISE                0.10
ROUTER_OVERDO                     0.0
ROUTER_SEED                       912345678

############################################################
# configuration file keys for better performance & results:

ROUTER_DISPLAY_PATHS           0         # excessive output = 1
ROUTER_INTERNAL_PLAN_SIZE      2000      # default = 400
ROUTER_MAX_NODES_EXAMINED      5000000   # default = 400000
ROUTER_MESSAGE_LEVEL           1         # severe=0, everything=1
ROUTER_NUMBER_THREADS          2         # per box
LOG_ROUTING                    1         # for speed calc
ROUTER_MAX_LEG_LENGTH          86400     # max 24 hour legs = basically unlimited
# This should limit route hopping without impacting route generation:
ROUTER_GET_ON_TRANSIT_DELAY    0
ROUTER_GET_OFF_TRANSIT_DELAY   60

############################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE        Activity_Location.20010806.tbl
NET_DETECTOR_TABLE                 Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE        Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                     Link.20011119.tbl
NET_NODE_TABLE                     Node.20010806.tbl
NET_PARKING_TABLE                  Parking.20010806.tbl
NET_PHASING_PLAN_TABLE             Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE              Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE             Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE       Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE          Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE        Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE              $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
# new configuration file key from Paula:

ROUTER_MAX_TRIP_TIME               172800
```

6.1.13 auto-C.cfg

```
########################################################################

# This file has Collator keys for creating an iteration database to be
# converted by ItdbByTour.pl into a "by-tour" database.  -JPS

########################################################################

TRANSIMS_ROOT                   /home/transims/CaseStudy3/scenarios/allstr/
#CONFIG_DEFAULT_FILE            $TRANSIMS_ROOT/allstr.cfg

NET_DIRECTORY                   $TRANSIMS_ROOT/network
NET_NODE_TABLE                  Node.tbl
NET_LINK_TABLE                  Link.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.tbl
NET_PARKING_TABLE               Parking.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.tbl
NET_TIMING_PLAN_TABLE           Timing_Plan.tbl
NET_SPEED_TABLE                 Speed.tbl
NET_LANE_USE_TABLE              Lane_Use.tbl
NET_TRANSIT_STOP_TABLE          Transit_Stop.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.tbl
NET_DETECTOR_TABLE              Detector.tbl
NET_TURN_PROHIBITION_TABLE      Turn_Prohibition.tbl
NET_BARRIER_TABLE               Barrier.tbl
NET_ACTIVITY_LOCATION_TABLE     Activity_Location.tbl
NET_PROCESS_LINK_TABLE          Process_Link.tbl

ACT_HOME_ACTIVITY_TYPE          0
ACT_WORK_ACTIVITY_TYPE          1
ACT_SCHOOL_ACTIVITY_TYPE        7

ACT_ANCHOR_ACTIVITY_TYPE_1      0      # Home
ACT_ANCHOR_ACTIVITY_TYPE_2      1      # Work
ACT_ANCHOR_ACTIVITY_TYPE_3      7      # School
ACT_ANCHOR_ACTIVITY_TYPE_4      8      # College

#################################################################
# configuration file keys for parallel runs:

#ACT_POPULATION_FILE      $TRANSIMS_ROOT/population/pop_converted
ACT_POPULATION_FILE       $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/sample4.pop

TRANSIT_ROUTE_FILE        $TRANSIMS_ROOT/network/Transit_Route.tbl

#################################################################

#SEL_POPULATION_FILE      $TRANSIMS_ROOT/population/pop_located
SEL_POPULATION_FILE       $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/sample4.pop
VEHICLE_FILE              $TRANSIMS_ROOT/vehicle/vehicles.all
SEL_PLAN_FILE             $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/auto.plans
SEL_ACTIVITY_FILE         $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/auto.act
#SEL_EVENT_FILE           $TRANSIMS_ROOT/output/anomaly.offplan.trv

SEL_ITDB_FILE             $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/auto

SEL_MESSAGE_LEVEL         0
SEL_NO_ITDB_INDEX         1

#################################################################
# Data gathered by the Collator for feedback
```

```
# Pop data
SEL_USE_RHHINC           1
SEL_USE_AGE              1

# Act data
SEL_USE_START_ACT_LOCATION        1
SEL_USE_END_ACT_TYPE              1
SEL_USE_END_MODE_PREF             1
SEL_USE_END_ACT_LOCATION          1
SEL_USE_END_OTHER_PARTICIPANTS    1
SEL_USE_END_DUR_UB                1
SEL_USE_END_DUR_LB                1
SEL_USE_DRIVER_ID                 1

# Activity Location Table data
SEL_USE_START_ACT_USER_DATA       Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8
SEL_USE_END_ACT_USER_DATA         Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8

# Route data
SEL_USE_EUCLID         1
SEL_USE_DURATION       1
SEL_USE_MODE_STRING    1

# event data
#SEL_USE_OFF_PLAN      1

############################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE       Activity_Location.20010806.tbl
NET_DETECTOR_TABLE                Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE       Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                    Link.20011119.tbl
NET_NODE_TABLE                    Node.20010806.tbl
NET_PARKING_TABLE                 Parking.20010806.tbl
NET_PHASING_PLAN_TABLE            Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE             Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE            Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE      Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE         Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE       Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE             $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
```

6.1.14 RemoveSubtours-C.cfg

```
########################################################################

# This file has Collator keys for creating an iteration database to be
# converted by ItdbByTour.pl into a "by-tour" database.  -JPS

########################################################################

TRANSIMS_ROOT                     /home/transims/CaseStudy3/scenarios/allstr/
#CONFIG_DEFAULT_FILE              $TRANSIMS_ROOT/allstr.cfg

NET_DIRECTORY                     $TRANSIMS_ROOT/network
NET_NODE_TABLE                    Node.tbl
NET_LINK_TABLE                    Link.tbl
NET_POCKET_LANE_TABLE             Pocket_Lane.tbl
NET_PARKING_TABLE                 Parking.tbl
NET_LANE_CONNECTIVITY_TABLE       Lane_Connectivity.tbl
NET_UNSIGNALIZED_NODE_TABLE       Unsignalized_Node.tbl
NET_SIGNALIZED_NODE_TABLE         Signalized_Node.tbl
NET_PHASING_PLAN_TABLE            Phasing_Plan.tbl
NET_TIMING_PLAN_TABLE             Timing_Plan.tbl
NET_SPEED_TABLE                   Speed.tbl
NET_LANE_USE_TABLE                Lane_Use.tbl
NET_TRANSIT_STOP_TABLE            Transit_Stop.tbl
NET_SIGNAL_COORDINATOR_TABLE      Signal_Coordinator.tbl
NET_DETECTOR_TABLE                Detector.tbl
NET_TURN_PROHIBITION_TABLE        Turn_Prohibition.tbl
NET_BARRIER_TABLE                 Barrier.tbl
NET_ACTIVITY_LOCATION_TABLE       Activity_Location.tbl
NET_PROCESS_LINK_TABLE            Process_Link.tbl

ACT_HOME_ACTIVITY_TYPE            0
ACT_WORK_ACTIVITY_TYPE            1
ACT_SCHOOL_ACTIVITY_TYPE          7

ACT_ANCHOR_ACTIVITY_TYPE_1        0     # Home
ACT_ANCHOR_ACTIVITY_TYPE_2        1     # Work
ACT_ANCHOR_ACTIVITY_TYPE_3        7     # School
ACT_ANCHOR_ACTIVITY_TYPE_4        8     # College

############################################################
#configuration file keys for parallel runs:

#ACT_POPULATION_FILE       $TRANSIMS_ROOT/population/pop_converted
ACT_POPULATION_FILE        $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/sample4.pop

TRANSIT_ROUTE_FILE         $TRANSIMS_ROOT/network/Transit_Route.tbl

############################################################

#SEL_POPULATION_FILE       $TRANSIMS_ROOT/population/pop_located
SEL_POPULATION_FILE        $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/sample4.pop
VEHICLE_FILE               $TRANSIMS_ROOT/vehicle/vehicles.all
#SEL_PLAN_FILE             $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225  plans/R-MF0.1
SEL_ACTIVITY_FILE          $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/sample4.act
#SEL_EVENT_FILE            $TRANSIMS_ROOT/output/anomaly.offplan.trv

SEL_ITDB_FILE              $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/sample4

SEL_MESSAGE_LEVEL          0
SEL_NO_ITDB_INDEX          1

############################################################
# Data gathered by the Collator for feedback
```

```
# Pop data
SEL_USE_RHHINC   1
SEL_USE_AGE      1

# Act data
SEL_USE_START_ACT_LOCATION         1
SEL_USE_END_ACT_TYPE               1
SEL_USE_END_MODE_PREF              1
SEL_USE_END_ACT_LOCATION           1
SEL_USE_END_OTHER_PARTICIPANTS     1
SEL_USE_END_DUR_UB                 1
SEL_USE_END_DUR_LB                 1
SEL_USE_DRIVER_ID                  1

# Activity Location Table data
SEL_USE_START_ACT_USER_DATA      Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8
SEL_USE_END_ACT_USER_DATA        Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8

# Route data
SEL_USE_EUCLID             1
SEL_USE_DURATION           1
SEL_USE_MODE_STRING        1

# event data
#SEL_USE_OFF_PLAN          1

############################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE        Activity_Location.20010806.tbl
NET_DETECTOR_TABLE                 Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE        Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                     Link.20011119.tbl
NET_NODE_TABLE                     Node.20010806.tbl
NET_PARKING_TABLE                  Parking.20010806.tbl
NET_PHASING_PLAN_TABLE             Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE              Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE             Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE       Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE          Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE        Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE              $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
```

6.1.15 final2-R.cfg

```
TRANSIMS_ROOT              /home/transims/CaseStudy3/scenarios/allstr

CONFIG_DEFAULT_FILE         $TRANSIMS_ROOT/allstr.cfg

############################################################
# configuration file keys for ONDN re-route run:

ACTIVITY_FILE                     $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final2.act
#ROUTER_LINK_DELAY_FILE           $TRANSIMS_ROOT/feedback/mode/ONDN/summary.tim

ROUTER_PROBLEM_FILE               $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final2.Router.problems
PLAN_FILE                         $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final2.plans
ROUTER_COMPLETED_HOUSEHOLD_FILE   $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final2.completed-HH

ROUTER_DELAY_NOISE                0.10
ROUTER_OVERDO                     0.0
ROUTER_SEED                       912345678

############################################################
# configuration file keys for better performance & results:

ROUTER_DISPLAY_PATHS           0         # excessive output = 1
ROUTER_INTERNAL_PLAN_SIZE      2000      # default = 400
ROUTER_MAX_NODES_EXAMINED      5000000   # default = 400000
ROUTER_MESSAGE_LEVEL           1         # severe=0, everything=1
ROUTER_NUMBER_THREADS          2         # per box
LOG_ROUTING                    1         # for speed calc
ROUTER_MAX_LEG_LENGTH          86400     # max 24 hour legs = basically unlimited
# This should limit route hopping without impacting route generation:
ROUTER_GET_ON_TRANSIT_DELAY    240
ROUTER_GET_OFF_TRANSIT_DELAY   240

############################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE       Activity_Location.20010806.tbl
NET_DETECTOR_TABLE                Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE       Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                    Link.20011119.tbl
NET_NODE_TABLE                    Node.20010806.tbl
NET_PARKING_TABLE                 Parking.20010806.tbl
NET_PHASING_PLAN_TABLE            Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE             Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE            Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE      Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE         Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE       Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE             $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
# new configuration file key from Paula:

ROUTER_MAX_TRIP_TIME    172800
```

6.1.16 final1-C.cfg

```
########################################################################

# This file has Collator keys for creating an iteration database to be
# converted by ItdbByTour.pl into a "by-tour" database.  -JPS

########################################################################

TRANSIMS_ROOT                   /home/transims/CaseStudy3/scenarios/allstr/
#CONFIG_DEFAULT_FILE            $TRANSIMS_ROOT/allstr.cfg

NET_DIRECTORY                   $TRANSIMS_ROOT/network
NET_NODE_TABLE                  Node.tbl
NET_LINK_TABLE                  Link.tbl
NET_POCKET_LANE_TABLE           Pocket_Lane.tbl
NET_PARKING_TABLE               Parking.tbl
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity.tbl
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node.tbl
NET_SIGNALIZED_NODE_TABLE       Signalized_Node.tbl
NET_PHASING_PLAN_TABLE          Phasing_Plan.tbl
NET_TIMING_PLAN_TABLE           Timing_Plan.tbl
NET_SPEED_TABLE                 Speed.tbl
NET_LANE_USE_TABLE              Lane_Use.tbl
NET_TRANSIT_STOP_TABLE          Transit_Stop.tbl
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator.tbl
NET_DETECTOR_TABLE              Detector.tbl
NET_TURN_PROHIBITION_TABLE      Turn_Prohibition.tbl
NET_BARRIER_TABLE               Barrier.tbl
NET_ACTIVITY_LOCATION_TABLE     Activity_Location.tbl
NET_PROCESS_LINK_TABLE          Process_Link.tbl

ACT_HOME_ACTIVITY_TYPE          0
ACT_WORK_ACTIVITY_TYPE          1
ACT_SCHOOL_ACTIVITY_TYPE        7

ACT_ANCHOR_ACTIVITY_TYPE_1      0     # Home
ACT_ANCHOR_ACTIVITY_TYPE_2      1     # Work
ACT_ANCHOR_ACTIVITY_TYPE_3      7    # School
ACT_ANCHOR_ACTIVITY_TYPE_4      8     # College

#################################################################
# configuration file keys for parallel runs:

#ACT_POPULATION_FILE    $TRANSIMS_ROOT/population/pop_converted
ACT_POPULATION_FILE     $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/application/sample4.pop

TRANSIT_ROUTE_FILE      $TRANSIMS_ROOT/network/Transit_Route.tbl

#################################################################

#SEL_POPULATION_FILE   $TRANSIMS_ROOT/population/pop_located
SEL_POPULATION_FILE    $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/application/sample4.pop
VEHICLE_FILE           $TRANSIMS_ROOT/vehicle/vehicles.all
SEL_PLAN_FILE          $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final1.plans
SEL_ACTIVITY_FILE      $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final1.act
#SEL_EVENT_FILE        $TRANSIMS_ROOT/output/anomaly.offplan.trv

SEL_ITDB_FILE          $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final1

SEL_MESSAGE_LEVEL      0
SEL_NO_ITDB_INDEX      1

#################################################################
# Data gathered by the Collator for feedback
```

```
# Pop data
SEL_USE_RHHINC    1
SEL_USE_AGE       1

# Act data
SEL_USE_START_ACT_LOCATION        1
SEL_USE_END_ACT_TYPE              1
SEL_USE_END_MODE_PREF             1
SEL_USE_END_ACT_LOCATION          1
SEL_USE_END_OTHER_PARTICIPANTS    1
SEL_USE_END_DUR_UB                1
SEL_USE_END_DUR_LB                1
SEL_USE_DRIVER_ID                 1

# Activity Location Table data
SEL_USE_START_ACT_USER_DATA    Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8
SEL_USE_END_ACT_USER_DATA      Tran_Dist; PARKING_ZN; URBAN_TYPE; River_Zone; Zones_8

# Route data
SEL_USE_EUCLID          1
SEL_USE_DURATION        1
SEL_USE_MODE_STRING     1

# event data
#SEL_USE_OFF_PLAN       1

############################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE        Activity_Location.20010806.tbl
NET_DETECTOR_TABLE                 Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE        Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                     Link.20011119.tbl
NET_NODE_TABLE                     Node.20010806.tbl
NET_PARKING_TABLE                  Parking.20010806.tbl
NET_PHASING_PLAN_TABLE             Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE              Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE             Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE       Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE          Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE        Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE              $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
```

6.1.17 final0-A.cfg

```
TRANSIMS_ROOT            /home/transims/CaseStudy3/scenarios/allstr

CONFIG_DEFAULT_FILE      $TRANSIMS_ROOT/allstr.cfg

############################################################
# recent corrections:

ACT_TRAVEL_TIMES_FILE              $TRANSIMS_ROOT/activity/traveltime_050101
ACT_TRAVEL_TIME_FUNCTION_MODES     1;2;3;4;5;6;7

ACT_SURVEY_ACTIVITY_FILE           $TRANSIMS_ROOT/data/survey_activities
ACT_SURVEY_WEIGHTS_FILE            $TRANSIMS_ROOT/data/survey_weights_25
#ACT_TRAVEL_TIME_INTERVALS_FILE    $TRANSIMS_ROOT/activity/time_intervals

LOG_ACT                            1

############################################################
# configuration file keys for ONDN activity feedback run:

ACTIVITY_FILE        $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/application/transit.act
ACT_FEEDBACK_FILE    $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final0.fdbk

ACT_PARTIAL_OUTPUT   $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final0.partial-act
ACT_PROBLEM_FILE     $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final0.ActivityRegenerator.problems
ACT_LOG_FILE         $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/final0.ActivityRegenerator.log

############################################################
# rockhopper changes:

# 24 hours
ROUTER_MAX_LEG_LENGTH            86400

# used for park and ride
ROUTER_MAX_NODES_EXAMINED        5000000

NET_ACTIVITY_LOCATION_TABLE      Activity_Location.20010806.tbl
NET_DETECTOR_TABLE               Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE      Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                   Link.20011119.tbl
NET_NODE_TABLE                   Node.20010806.tbl
NET_PARKING_TABLE                Parking.20010806.tbl
NET_PHASING_PLAN_TABLE           Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE            Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE           Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE     Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE        Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE      Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE            $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
```

6.1.18 test-R.cfg

```
TRANSIMS_ROOT /home/transims/CaseStudy3/scenarios/allstr

CONFIG_DEFAULT_FILE  $TRANSIMS_ROOT/allstr.cfg

############################################################
# configuration file keys for ONDN re-route run:

ACTIVITY_FILE                     $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.act
#ROUTER_LINK_DELAY_FILE           $TRANSIMS_ROOT/feedback/mode/ONDN/summary.tim

ROUTER_PROBLEM_FILE               $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/test.Router.problems
PLAN_FILE                         $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/test.plans
ROUTER_COMPLETED_HOUSEHOLD_FILE   $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/test.completed-HH

ROUTER_DELAY_NOISE                0.10
ROUTER_OVERDO                     0.0
ROUTER_SEED                       912345678

############################################################
# configuration file keys for better performance & results:

ROUTER_DISPLAY_PATHS          0          # excessive output = 1
ROUTER_INTERNAL_PLAN_SIZE     2000       # default = 400
ROUTER_MAX_NODES_EXAMINED     5000000    # default = 400000
ROUTER_MESSAGE_LEVEL          1          # severe=0, everything=1
ROUTER_NUMBER_THREADS         2          # per box
LOG_ROUTING                   1          # for speed calc
ROUTER_MAX_LEG_LENGTH         86400      # max 24 hour legs = basically unlimited
# This should limit route hopping without impacting route generation:
ROUTER_GET_ON_TRANSIT_DELAY   240
ROUTER_GET_OFF_TRANSIT_DELAY  240

############################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE       Activity_Location.20010806.tbl
NET_DETECTOR_TABLE                Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE       Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                    Link.20011119.tbl
NET_NODE_TABLE                    Node.20010806.tbl
NET_PARKING_TABLE                 Parking.20010806.tbl
NET_PHASING_PLAN_TABLE            Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE             Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE            Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE      Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE         Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE       Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE             $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
# new configuration file key from Paula:

ROUTER_MAX_TRIP_TIME              172800
```

6.1.19 transit-R.cfg

```
TRANSIMS_ROOT               /home/transims/CaseStudy3/scenarios/allstr

CONFIG_DEFAULT_FILE        $TRANSIMS_ROOT/allstr.cfg

############################################################
# configuration file keys for ONDN re-route run:

ACTIVITY_FILE                     $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.act
#ROUTER_LINK_DELAY_FILE           $TRANSIMS_ROOT/feedback/mode/ONDN/summary.tim

ROUTER_PROBLEM_FILE               $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.Router.problems
PLAN_FILE                         $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.plans
ROUTER_COMPLETED_HOUSEHOLD_FILE   $TRANSIMS_ROOT/feedback/mode/ONDN/Test20020225/transit.completed-HH

ROUTER_DELAY_NOISE                0.10
ROUTER_OVERDO                     0.0
ROUTER_SEED                       912345678

############################################################
# configuration file keys for better performance & results:

ROUTER_DISPLAY_PATHS           0           # excessive output = 1
ROUTER_INTERNAL_PLAN_SIZE      2000        # default = 400
ROUTER_MAX_NODES_EXAMINED      5000000     # default = 400000
ROUTER_MESSAGE_LEVEL           1           # severe=0, everything=1
ROUTER_NUMBER_THREADS          2           # per box
LOG_ROUTING                    1           # for speed calc
ROUTER_MAX_LEG_LENGTH          86400       # max 24 hour legs = basically unlimited
# This should limit route hopping without impacting route generation:
ROUTER_GET_ON_TRANSIT_DELAY    0
ROUTER_GET_OFF_TRANSIT_DELAY   60

############################################################
# rockhopper changes:

NET_ACTIVITY_LOCATION_TABLE       Activity_Location.20010806.tbl
NET_DETECTOR_TABLE                Detector.20010921.tbl
NET_LANE_CONNECTIVITY_TABLE       Lane_Connectivity.20011119.tbl
NET_LINK_TABLE                    Link.20011119.tbl
NET_NODE_TABLE                    Node.20010806.tbl
NET_PARKING_TABLE                 Parking.20010806.tbl
NET_PHASING_PLAN_TABLE            Phasing_Plan.20010921.tbl
NET_POCKET_LANE_TABLE             Pocket_Lane.20010921.tbl
NET_PROCESS_LINK_TABLE            Process_Link_061901.tbl
NET_SIGNAL_COORDINATOR_TABLE      Signal_Coordinator.20010806.tbl
NET_SIGNALIZED_NODE_TABLE         Signalized_Node.20010921.tbl
NET_UNSIGNALIZED_NODE_TABLE       Unsignalized_Node.20011119.tbl

TRANSIT_SCHEDULE_FILE             $TRANSIMS_ROOT/network/Transit_Schedule.20011113.tbl

############################################################
# new configuration file key from Paula:

ROUTER_MAX_TRIP_TIME              172800
```

6.2 Scripts

Note: Long code lines that do not fit completely on one line are shown in *italics* and continued on to the next line.

6.2.1 run.csh

```
#! /bin/csh

##################################
# Step 1: definitions only

cd /home/transims/CaseStudy3/scenarios/allstr/feedback/mode/ONDN/Test20020225


##################################
# Step 2: AS7_ONDN.w.1.1.tdb

cp ../itdb/init/AS7.it.gz .
gunzip AS7.it.gz

./scripts/ItdbByTour.pl AS7.it AS7.tdb

./scripts/FilterONDN.pl AS7.tdb AS7_ONDN.tdb

./scripts/Stratify.pl AS7_ONDN.tdb AS7_ONDN
mv AS7_ONDN.w.1.1  AS7_ONDN.w.1.1.tdb
rm AS7_ONDN.?.?.?


##################################
# Step 3: sample1.tdb

./scripts/FilterOther.pl AS7_ONDN.w.1.1.tdb AS7_ONDN.w.1.1_s=0_age_X.tdb

head -1 AS7_ONDN.w.1.1_s=0_age_X.tdb >! sample0.tdb
./scripts/Select-Random.pl 24823341 0.019 AS7_ONDN.w.1.1_s=0_age_X.tdb tmp1.tdb tmp2.tdb
cat tmp1.tdb >> sample0.tdb

# Remove school & college tours
gawk 'NR==1{print} NR>1{if($31==1)print}' sample0.tdb >! tmp3.tdb
# remove travelers using mode 8
gawk '{if($2!=520083 && $2!=523583 && $2!=529488 && $2!=741155 && $2!=743571 &&
$2!=527237 && $2!=534805 && $2!=536018)print}' tmp3.tdb >! tmp4.tdb
# reduce number to 100 +1 bike trip:
gawk '{if(NR!=85)print}' tmp4.tdb >! tmp5.tdb
# remove nightshift worker
gawk '{if($2!=506629)print}' tmp5.tdb >! sample3.tdb


##################################
# Step 4: sample1.pop, sample1.act

./scripts/Match_Acts.pl sample3.tdb
/home/transims/CaseStudy3/scenarios/allstr/activity/AS7nt tmp1.act
# clean out extra home activities
gawk '{if($2!=518380 || $3>4)print}' tmp1.act > tmp2.act

# re-time & re-number
./scripts/FixActs.pl tmp2.act > tmp3.act
gawk '{if($2==527178 && $3==3){$10=36;$11=36;$14=36-$7;$15=36-$6}print}' OFS="\t"
tmp3.act > sample3.act
```

```
./scripts/Match_Pop.pl sample3.tdb
/home/transims/CaseStudy3/scenarios/allstr/population/pop_located sample3.pop


##################################
# Step 5: no-subtours.act, transit.act

# remove sub-tours
/home/transims/CaseStudy3/bin/Collator config/RemoveSubtours-C.cfg
gawk 'NR>2{if($4!=0)print $1,$7,"MS 8"}' FS="," sample3.000.it | sort -n | uniq >! no-
subtours.fdbk

# regenerate acts
/home/transims/CaseStudy3/bin/ActivityRegenerator config/RemoveSubtours-A.cfg

# merge with previous
$TRANSIMS_HOME/bin/MergeIndices no-subtours.hh.idx sample3.act.hh.idx no-
subtours.partial-act.hh.idx
$TRANSIMS_HOME/bin/IndexDefrag no-subtours.hh.idx no-subtours.act
rm no-subtours no-subtours.hh.idx

# create tour database with correct numbering
./scripts/ItdbByTour.pl sample3.000.it sample3-renumb.tdb

# all X modes use generalized transit
./scripts/ReModeTrans.pl sample3-renumb.tdb transit.HH transit.fdbk

# regenerate acts
/home/transims/CaseStudy3/bin/ActivityRegenerator config/ReModeTrans-A.cfg

# change all first activities to mode 3 (aesthetic reasons only)
mv transit.act tmp4.act
gawk '{if($18==2)$18=1;print}' OFS="\t" tmp4.act > transit.act


##################################
# Step 6:

# route transit.act
/home/transims/CaseStudy3/bin/Router config/transit-R.cfg >&! transit.Router.log

# collate
/home/transims/CaseStudy3/bin/Collator config/transit-C.cfg

# tour-based
./scripts/ItdbByTour.pl transit.000.it transit.tdb

# data file
gawk 'NR==1{print "HH\tTrav\tAct\tIncome\tParking\tT-Time\tT-AnchorMod\tDistance"}
NR>1{print $1,$2,$6,$3,$28,$10,$31,$11}' OFS="\t" transit.tdb >! transit.dat


##################################
# Step 6a: attempt to change Router to slow transit over walk
# Could change on/off transit delays, or walking speed.

# route transit.act
/home/transims/CaseStudy3/bin/Router config/test-R.cfg >&! test.Router.log

# collate
/home/transims/CaseStudy3/bin/Collator config/test-C.cfg

# tour-based
./scripts/ItdbByTour.pl test.000.it test.tdb

# data file
gawk 'NR==1{print "HH\tTrav\tAct\tIncome\tParking\tT-Time\tT-AnchorMod\tDistance"}
NR>1{print $1,$2,$6,$3,$28,$10,$31,$11}' OFS="\t" test.tdb >! test.dat
```

```
###################################
# Step 7: count router's generalized transit split into walk or transit

gawk '{print $7}' transit.dat | sort -n | uniq -c | gawk 'NR==1{print"AnchMod\tFraction"}
NR>1{print $2,$1/100}' OFS="\t"


###################################
# Step 8: Eq.14 is negative;
#         1% chance sample drawn from METRO's target distr.


###################################
# Step 9: auto.act, auto.plans, auto.tdb, auto.dat

# replace generalized transit with auto
gawk 'NR>1{print $1,$6,"M 2 3"}' transit.tdb >! auto.fdbk

# regenerate acts
/home/transims/CaseStudy3/bin/ActivityRegenerator config/ReModeAuto-A.cfg

# route auto.act
/home/transims/CaseStudy3/bin/Router config/auto-R.cfg >&! auto.Router.log

# collate
/home/transims/CaseStudy3/bin/Collator config/auto-C.cfg

# tour-based
./scripts/ItdbByTour.pl auto.000.it auto.tdb

# data file
gawk 'NR==1{print "HH\tTrav\tAct\tIncome\tParking\tA-Time\tA-AnchorMod\tDistance"}
NR>1{print $1,$2,$6,$3,$28,$10,$31,$11}' OFS="\t" auto.tdb >! auto.dat


###################################
# Step 10: combined.w.dat, combined.t.dat

# get walk tours from test.dat (exclude bike trip)
gawk 'NR==1{print} NR>1{if($7==1 && $2!=534042)print}' test.dat >! transit.w.dat

# get transit tours from test.dat
gawk 'NR==1{print} NR>1{if($7!=1)print}' test.dat >! transit.t.dat

# OR
# get transit tours from transit.dat (match with test.dat)
head -1 transit.dat >! transit.t.dat
foreach t (`gawk 'NR>1{if($7!=1)print $2}' test.dat`)
gawk -v T=$t '{if($2==T)print}' transit.dat >> transit.t.dat
end

# combine, add parking cost (NOTE: script is hard-wired for fields)
./scripts/CombineDat.pl transit.w.dat auto.dat >! combined.w.dat
./scripts/CombineDat.pl transit.t.dat auto.dat >! combined.t.dat


###################################
# Step 11: alpha_w = 0.0000033, alpha_t = 0.00004, auto_rate = $0.03/km

# calibrate
gawk -v a=0.0000033 -v r=0.03
'NR>1{Cw=a*log($4+1.01)*$6;Ca=a*log($4+1.01)*$9+$11+r*$8/1000;if(Cw<Ca)sum++;} END{print
sum}' combined.w.dat

gawk -v a=0.00004 -v r=0.0
'NR>1{Ct=a*log($4+1.01)*$6+1;Ca=a*log($4+1.01)*$9+$11+r*$8/1000;if(Ct<Ca)sum++;}
END{print sum}' combined.t.dat
# OR
```

```
gawk -v a=0.00004 -v r=0.03
'NR>1{Ct=a*log($4+1.01)*$6+1;Ca=a*log($4+1.01)*$9+$11+r*$8/1000;if(Ct<Ca)sum++;}
END{print sum}' combined.t.dat

# then make cost databases
gawk -v a=0.0000033 -v r=0.03 'NR==1{print $1,$3,$7,"T-Cost",$10,"A-Cost"}
NR>1{Cw=a*log($4+1.01)*$6;Ca=a*log($4+1.01)*$9+$11+r*$8/1000;print $1,$3,$7,Cw,$10,Ca}'
OFS="\t" combined.w.dat >! combined.w.cdb

gawk -v a=0.00004 -v r=0.03 'NR==1{print $1,$3,$7,"T-Cost",$10,"A-Cost"}
NR>1{Ct=a*log($4+1.01)*$6+1;Ca=a*log($4+1.01)*$9+$11+r*$8/1000;print $1,$3,$7,Ct,$10,Ca}'
OFS="\t" combined.t.dat >! combined.t.cdb

###################################

# clean
mkdir calibration
mv *.* calibration/


###################################
###################################
# Step 12: final0.act, final0.plans, final0.tdb, final0.dat

# basically repeat steps 3-9

head -1 calibration/AS7_ONDN.w.1.1_s=0_age_X.tdb >! sample0.tdb
./scripts/Select-Random.pl 2482341 0.0205 calibration/AS7_ONDN.w.1.1_s=0_age_X.tdb
tmp1.tdb
cat tmp1.tdb >> sample0.tdb

# Remove school & college tours
gawk 'NR==1{print} NR>1{if($31==1)print}' sample0.tdb >! tmp3.tdb
# remove travelers using mode 8 on primary anchor
gawk '{if($2!=528530 && $2!=535111 && $2!=681479 && $2!=682445 && $2!=741155)print}'
tmp3.tdb >! tmp4.tdb
# remove travelers using mode 8 elsewhere
gawk '{if($2!=528127 && $2!=534730 && $2!=672569 && $2!=672606)print}' tmp4.tdb >!
tmp5.tdb
# remove duplicate traveler
gawk 'NR==1{print;first=1;} NR>1{if($2==515965){if(first==1)print;first=-1}else{print}}'
tmp5.tdb >! sample4.tdb

###################################

./scripts/Match_Acts.pl sample4.tdb
/home/transims/CaseStudy3/scenarios/allstr/activity/AS7nt tmp1.act
# clean out extra home activities
gawk '{if($2!=529320 || $3<4)print}' tmp1.act > tmp2.act

# re-time & re-number
./scripts/FixActs.pl tmp2.act > sample4.act

./scripts/Match_Pop.pl sample4.tdb
/home/transims/CaseStudy3/scenarios/allstr/population/pop_located sample4.pop

###################################

# remove sub-tours
/home/transims/CaseStudy3/bin/Collator config/RemoveSubtours-C.cfg
gawk 'NR>2{if($4!=0)print $1,$7,"MS 8"}' FS="," sample4.000.it | sort -n | uniq >! no-
subtours.fdbk
rm sample4.idx

# regenerate acts
/home/transims/CaseStudy3/bin/ActivityRegenerator config/RemoveSubtours-A.cfg

# merge with previous
$TRANSIMS_HOME/bin/MergeIndices no-subtours.hh.idx sample4.act.hh.idx no-
subtours.partial-act.hh.idx
$TRANSIMS_HOME/bin/IndexDefrag no-subtours.hh.idx no-subtours.act
```

```
rm no-subtours.partial-act.*.idx no-subtours.hh.idx no-subtours.fdbk.*.idx

# create tour database with correct numbering
./scripts/ItdbByTour.pl sample4.000.it sample4-renumb.tdb

# all X modes use generalized transit
./scripts/ReModeTrans.pl sample4-renumb.tdb transit.HH transit.fdbk

# regenerate acts
/home/transims/CaseStudy3/bin/ActivityRegenerator config/ReModeTrans-A.cfg

# change all first activities to mode 1 (aesthetic reasons only)
mv transit.act tmp4.act
gawk '{if($18==2)$18=1;print}' OFS="\t" tmp4.act > transit.act

###################################

# route transit.act
/home/transims/CaseStudy3/bin/Router config/test-R.cfg >&! test.Router.log

# collate
/home/transims/CaseStudy3/bin/Collator config/test-C.cfg

# tour-based
./scripts/ItdbByTour.pl test.000.it test.tdb

# data file
gawk 'NR==1{print "HH\tTrav\tAct\tIncome\tParking\tT-Time\tT-AnchorMod\tDistance"}
NR>1{print $1,$2,$6,$3,$28,$10,$31,$11}' OFS="\t" test.tdb >! test.dat

###################################

gawk '{print $7}' test.dat | sort -n | uniq -c | gawk 'NR==1{print"AnchMod\tFraction"}
NR>1{print $2,$1/100}' OFS="\t"

###################################

# replace generalized transit with auto
gawk 'NR>1{print $1,$6,"M 2 3"}' test.tdb >! auto.fdbk

# regenerate acts
/home/transims/CaseStudy3/bin/ActivityRegenerator config/ReModeAuto-A.cfg

# route auto.act
/home/transims/CaseStudy3/bin/Router config/auto-R.cfg >&! auto.Router.log

# collate
/home/transims/CaseStudy3/bin/Collator config/auto-C.cfg

# tour-based
./scripts/ItdbByTour.pl auto.000.it auto.tdb

# data file
gawk 'NR==1{print "HH\tTrav\tAct\tIncome\tParking\tA-Time\tA-AnchorMod\tDistance"}
NR>1{print $1,$2,$6,$3,$28,$10,$31,$11}' OFS="\t" auto.tdb >! auto.dat

###################################

# get walk tours from test.dat
gawk 'NR==1{print} NR>1{if($7==1)print}' test.dat >! transit.w.dat

# get transit tours from test.dat
gawk 'NR==1{print} NR>1{if($7!=1)print}' test.dat >! transit.t.dat

# OR
# get transit tours from transit.dat (match with test.dat)
head -1 transit.dat >! transit.t.dat
foreach t (`gawk 'NR>1{if($7!=1)print $2}' test.dat`)
gawk -v T=$t '{if($2==T)print}' transit.dat >> transit.t.dat
end
```

```
# combine, add parking cost (NOTE: script is hard-wired for fields)
./scripts/CombineDat.pl transit.w.dat auto.dat >! combined.w.dat
./scripts/CombineDat.pl transit.t.dat auto.dat >! combined.t.dat

##################################
##################################

# cost databases
gawk -v a=0.0000033 -v r=0.03 'NR==1{print $1,$3,$7,"T-Cost",$10,"A-Cost"}
NR>1{Cw=a*log($4+1.01)*$6;Ca=a*log($4+1.01)*$9+$11+r*$8/1000;print $1,$3,$7,Cw,$10,Ca}'
OFS="\t" combined.w.dat >! combined.w.cdb

gawk -v a=0.00004 -v r=0.03 'NR==1{print $1,$3,$7,"T-Cost",$10,"A-Cost"}
NR>1{Ct=a*log($4+1.01)*$6+1;Ca=a*log($4+1.01)*$9+$11+r*$8/1000;print $1,$3,$7,Ct,$10,Ca}'
OFS="\t" combined.t.dat >! combined.t.cdb

# clean
mkdir application
mv *.* application/

# guess modes
./bin/GuessMode 1 2 0.1 10 11 application/combined.w.cdb application/combined.w.cdb
final0.w.HH final0.w.fdbk final0.w.log
./bin/GuessMode 3 2 0.1 10 11 application/combined.t.cdb application/combined.t.cdb
final0.t.HH final0.t.fdbk final0.t.log

# make rail trip look like a regular transit trip
gawk '{if($3==5){$3=3}print}' OFS="\t" application/combined.t.cdb >! combined.t2.cdb
./bin/GuessMode 3 2 0.1 10 11 combined.t2.cdb combined.t2.cdb final0.t.HH final0.t.fdbk
final0.t.log

# summary of modes:
echo "HH\tAct\tT-mode\tA-prob" >! summary0.dat
gawk '{print $1,$2,"w",$3}' OFS="\t" final0.w.log >> summary0.dat
gawk '{print $1,$2,"t",$3}' OFS="\t" final0.t.log >> summary0.dat
gawk 'BEGIN{min=0;max=0} {sum+=$3;if($3!=0){max++};if($3==1){min++}} END{print
"expected/min/max walks: ",NR-sum,NR-max,NR-min}' final0.w.log
gawk 'BEGIN{min=0;max=0} {sum+=$3;if($3!=0){max++};if($3==1){min++}} END{print
"expected/min/max transit: ",NR-sum,NR-max,NR-min}' final0.t.log
gawk 'BEGIN{min=0;max=0} {sum+=$4;if($4!=0){max++};if($4==1){min++}} END{print
"expected/min/max auto: ",sum,min,max}' summary0.dat

# merge changes (Note: walks are actually implemented as general-transit)
cat final0.?.fdbk | sed 's:2 M 2 1:2 M 2 3:g' | sort -n > final0.fdbk
cat final0.?.HH | sort -n | uniq > final0.HH

# implement from transit.act
/home/transims/CaseStudy3/bin/ActivityRegenerator config/final0-A.cfg

# merge with transit.act
$TRANSIMS_HOME/bin/MergeIndices final0.hh.idx application/transit.act.hh.idx
final0.partial-act.hh.idx
$TRANSIMS_HOME/bin/IndexDefrag final0.hh.idx final0.act
rm final0.partial-act.*.idx final0.hh.idx final0.fdbk.*.idx application/transit.act.*.idx

# route final0.act
/home/transims/CaseStudy3/bin/Router config/final0-R.cfg >&! final0.Router.log

# collate
/home/transims/CaseStudy3/bin/Collator config/final0-C.cfg

# tour-based
./scripts/ItdbByTour.pl final0.000.it final0.tdb

# data file
gawk 'NR==1{print "HH\tTrav\tAct\tIncome\tParking\tTime\tAnchorMod\tDistance"} NR>1{print
$1,$2,$6,$3,$28,$10,$31,$11}' OFS="\t" final0.tdb >! final0.dat

# check if iteration is necessary (more than 4 transfers)
gawk '{print $30}' FS="," final0.000.it | sed 's:w::g' | sed 's:l:b:g' | sort | uniq -c
```

```
# if iteration were necessary, do it by adding "reversing" feedback
# commands for offending tours, then re-run the guess program with a
# different seed, and select one of those randomly to add to the
# correction feedback commands.  Act, Route, Collate, etc.


##################################
##################################
# Step 13: final1.act, final1.plans, final1.tdb, final1.dat,
#          final2.act, final2.plans, final2.tdb, final2.dat

# guess modes
./bin/GuessMode2 1 2 0.1 10 19 application/combined.w.cdb application/combined.w.cdb
final1.w.HH final1.w.fdbk final1.w.log
./bin/GuessMode2 3 2 0.1 10 19 combined.t2.cdb combined.t2.cdb final1.t.HH final1.t.fdbk
final1.t.log

# summary of modes:
echo "HH\tAct\tT-mode\tA-prob" >! summary1.dat
gawk '{print $1,$2,"w",$3}' OFS="\t" final1.w.log >> summary1.dat
gawk '{print $1,$2,"t",$3}' OFS="\t" final1.t.log >> summary1.dat
gawk 'BEGIN{min=0;max=0} {sum+=$3;if($3!=0){max++};if($3==1){min++}} END{print
"expected/min/max walks: ",NR-sum,NR-max,NR-min}' final1.w.log
gawk 'BEGIN{min=0;max=0} {sum+=$3;if($3!=0){max++};if($3==1){min++}} END{print
"expected/min/max transit: ",NR-sum,NR-max,NR-min}' final1.t.log
gawk 'BEGIN{min=0;max=0} {sum+=$4;if($4!=0){max++};if($4==1){min++}} END{print
"expected/min/max auto: ",sum,min,max}' summary1.dat

# merge changes (Note: walks are actually implemented as general-transit)
cat final1.?.fdbk | sed 's:2 M 2 1:2 M 2 3:g' | sort -n > final1.fdbk
cat final1.?.HH | sort -n | uniq > final1.HH

# implement from transit.act
/home/transims/CaseStudy3/bin/ActivityRegenerator config/final1-A.cfg

# merge with transit.act
$TRANSIMS_HOME/bin/MergeIndices final1.hh.idx application/transit.act.hh.idx
final1.partial-act.hh.idx
$TRANSIMS_HOME/bin/IndexDefrag final1.hh.idx final1.act
rm final1.partial-act.*.idx final1.hh.idx final1.fdbk.*.idx application/transit.act.*.idx

# route final1.act
/home/transims/CaseStudy3/bin/Router config/final1-R.cfg >&! final1.Router.log

# collate
/home/transims/CaseStudy3/bin/Collator config/final1-C.cfg

# tour-based
./scripts/ItdbByTour.pl final1.000.it final1.tdb

# data file
gawk 'NR==1{print "HH\tTrav\tAct\tIncome\tParking\tTime\tAnchorMod\tDistance"} NR>1{print
$1,$2,$6,$3,$28,$10,$31,$11}' OFS="\t" final1.tdb >! final1.dat

# check if iteration is necessary (more than 4 transfers)
gawk '{print $30}' FS="," final1.000.it | sed 's:w::g' | sed 's:l:b:g' | sort | uniq -c

##################################

# arbitrarily split by transit mode from calibration
tail +2 application/auto.dat >! tmp1.dat
./scripts/Select-Random.pl 7 0.56 tmp1.dat tmp.w.dat tmp.t.dat
gawk 'NR==1{print $_,"ParkingCost"}' OFS="\t" application/auto.dat >! auto.w.dat
gawk 'NR==1{print $_,"ParkingCost"}' OFS="\t" application/auto.dat >! auto.t.dat
./scripts/AddParkingCost.pl tmp.w.dat >> auto.w.dat
./scripts/AddParkingCost.pl tmp.t.dat >> auto.t.dat
rm tmp1.dat tmp.w.dat tmp.t.dat

# make cost databases
```

```
gawk -v a=0.0000033 -v r=0.03 'NR==1{print $1,$3,$7,"A-Cost"}
NR>1{Ca=a*log($4+1.01)*$6+$9+r*$8/1000;print $1,$3,$7,Ca}' OFS="\t" auto.w.dat >!
auto.w.cdb

gawk -v a=0.00004 -v r=0.03 'NR==1{print $1,$3,$7,"A-Cost"}
NR>1{Ca=a*log($4+1.01)*$6+$9+r*$8/1000;print $1,$3,$7,Ca}' OFS="\t" auto.t.dat >!
auto.t.cdb

# guess modes
./bin/GuessMode 1 2 0.1 10 11 calibration/combined.w.cdb auto.w.cdb final2.w.HH
final2.w.fdbk final2.w.log
./bin/GuessMode 3 2 0.1 10 11 calibration/combined.t.cdb auto.t.cdb final2.t.HH
final2.t.fdbk final2.t.log

# check distribution from which sample is drawn
gawk '{print $6,($6>$4)}' OFS="\t" calibration/combined.w.cdb | sort -n

# merge changes (Note: walks are actually implemented as general-transit)
cat final2.?.fdbk | sed 's:2 M 1 2:2 M 3 2:g' | sort -n > final2.fdbk
cat final2.?.HH | sort -n | uniq > final2.HH

# implement from auto.act
/home/transims/CaseStudy3/bin/ActivityRegenerator config/final2-A.cfg

# merge with auto.act
$TRANSIMS_HOME/bin/MergeIndices final2.hh.idx application/auto.act.hh.idx final2.partial-
act.hh.idx
$TRANSIMS_HOME/bin/IndexDefrag final2.hh.idx final2.act
rm final2.partial-act.*.idx final2.hh.idx final2.fdbk.*.idx application/auto.act.*.idx

# route final2.act
/home/transims/CaseStudy3/bin/Router config/final2-R.cfg >&! final2.Router.log

# collate
/home/transims/CaseStudy3/bin/Collator config/final2-C.cfg

# tour-based
./scripts/ItdbByTour.pl final2.000.it final2.tdb

# data file
gawk 'NR==1{print "HH\tTrav\tAct\tIncome\tParking\tTime\tAnchorMod\tDistance"} NR>1{print
$1,$2,$6,$3,$28,$10,$31,$11}' OFS="\t" final2.tdb >! final2.dat

# check if iteration is necessary (more than 4 transfers)
gawk '{print $30}' FS="," final2.000.it | sed 's:w::g' | sed 's:l:b:g' | sort | uniq -c
# check mode split
gawk '{print $7}' final2.dat | sort -n | uniq -c

###################################
```

6.2.2 FixActs.pl

```
#!/usr/bin/perl
# Solaris: /sw/Cvol/bin/perl

$delim = "\t";
$info = 0;

open(FILE1, $ARGV[0]) or die "Can't open file $ARGV[0].\n";
for($i=0;$i<$info;$i++) {$line = <FILE1>;if($i==$info-1){print $line;}}

$LastTrav = -1;

while(<FILE1>) {
    $line = $_; chop $line;
    @data = split($delim, $line);

    $Trav = $data[1];

    if( $LastTrav != $Trav ) {
        $ActID = 1;
        $data[13] = $data[9];
        $data[14] = $data[10];
        $data[5] = 0;
        $data[6] = 0;
        $Home = $data[20];
    }
    elsif( $Home == $data[20] ) {
        $ActID++;
        $data[13] = 36 - $data[6];
        $data[14] = 36 - $data[5];
        $data[9] = 36;
        $data[10] = 36;
    }
    else {
        $ActID++;
    }

    $data[2] = $ActID;

    print join("\t",@data),"\n";

    $LastTrav = $Trav;
}

close FILE1;
```

6.2.3 FilterONDN.pl

```
#! /usr/bin/perl -w

##########################################################################

# This takes a tour-based itdb as input and filters to include only
# non-shared, ONDN tours with traveler's age>15.

##########################################################################

$delim = "\t";
$heads = 1;

$TranDist = 1000; # maximum transit distance

# get command line arguments:
if($#ARGV+1 != 2) {
    print "\nusage: FilterONDN.pl TOURITDB NEWDB\n\n";
    print "   TOURITDB = input tour-based, tab delimited iteration database\n";
    print "   NEWDB = output as TOURITDB, but only selected tours\n\n";
    exit;
}
local($origitdb, $outitdb) = @ARGV;

# open input itdb and parse header
open(ORIGITDB, "$origitdb") || die "Failed to open $origitdb.";
# read header lines
for($i=0;$i<$heads;$i++) {
    $line = <ORIGITDB>;
}

# find fields automatically using header
chomp $line;
@data = split($delim, $line);
$i=0;
$AgeField = $ModeField = $SharField = $Tran1Field = $Tran2Field = -1;
while($data[$i]) {
    if($data[$i] eq "Age") {
        $AgeField = $i;
    }
    elsif($data[$i] eq "Mode") {
        $ModeField = $i;
    }
    elsif($data[$i] eq "Shared") {
        $SharField = $i;
    }
    elsif($data[$i] eq "HomeTranDist") {
        $Tran1Field = $i;
    }
    elsif($data[$i] eq "AnchorTranDist") {
        $Tran2Field = $i;
    }
    $i++;
}
# check for mandatory fields
$Missing = 0;
if($AgeField == -1) {
    print "Missing Age.\n";
    $Missing++;
}
if($ModeField == -1) {
    print "Missing Mode.\n";
    $Missing++;
}
```

```
if($SharField == -1) {
    print "Missing Shared.\n";
    $Missing++;
}
if($Tran1Field == -1) {
    print "Missing HomeTranDist.\n";
    $Missing++;
}
if($Tran2Field == -1) {
    print "Missing AnchorTranDist.\n";
    $Missing++;
}
if($Missing > 0) {exit;}


# open output files
open(NEWITDB, ">$outitdb") || die "Failed to open $outitdb.";

# print itdb header
print NEWITDB $line,"\n";

############################

# loop over lines
while ( <ORIGITDB> ) {
    $line = $_; chomp $line;
    @data = split($delim, $line);

    # not-shared; ONDN; age>15;
    if( $data[$Tran1Field]<$TranDist && $data[$Tran2Field]<$TranDist ) {

        # print to new itdb
        print NEWITDB $line,"\n";
    }
}

############################

close ORIGITDB;
close NEWITDB;

############################
exit;
```

6.2.4 FilterOther.pl

```
#! /usr/bin/perl -w

##########################################################################

# This takes a tour-based itdb as input and filters to include only
# non-shared, ONDN tours with traveler's age>15.

##########################################################################

$delim = "\t";
$heads = 1;

# get command line arguments:
if($#ARGV+1 != 2) {
    print "\nusage: FilterONDN.pl TOURITDB NEWDB\n\n";
    print "   TOURITDB = input tour-based, tab delimited iteration database\n";
    print "   NEWDB = output as TOURITDB, but only selected tours\n\n";
    exit;
}
local($origitdb, $outitdb) = @ARGV;

# open input itdb and parse header
open(ORIGITDB, "$origitdb") || die "Failed to open $origitdb.";
# read header lines
for($i=0;$i<$heads;$i++) {
    $line = <ORIGITDB>;
}

# find fields automatically using header
chomp $line;
@data = split($delim, $line);
$i=0;
$AgeField = $ModePrefField = $SharField = $Tran1Field = $Tran2Field = -1;
while($data[$i]) {
    if($data[$i] eq "Age") {
        $AgeField = $i;
    }
    elsif($data[$i] eq "ModePref") {
        $ModePrefField = $i;
    }
    elsif($data[$i] eq "Shared") {
        $SharField = $i;
    }
    elsif($data[$i] eq "HomeTranDist") {
        $Tran1Field = $i;
    }
    elsif($data[$i] eq "AnchorTranDist") {
        $Tran2Field = $i;
    }
    $i++;
}
# check for mandatory fields
$Missing = 0;
if($AgeField == -1) {
    print "Missing Age.\n";
    $Missing++;
}
if($ModePrefField == -1) {
    print "Missing ModePref.\n";
    $Missing++;
}
if($SharField == -1) {
    print "Missing Shared.\n";
    $Missing++;
}
```

```
if($Tran1Field == -1) {
    print "Missing HomeTranDist.\n";
    $Missing++;
}
if($Tran2Field == -1) {
    print "Missing AnchorTranDist.\n";
    $Missing++;
}
if($Missing > 0) {exit;}


# open output files
open(NEWITDB, ">$outitdb") || die "Failed to open $outitdb.";

# print itdb header
print NEWITDB $line,"\n";

############################

# loop over lines
while ( <ORIGITDB> ) {
    $line = $_; chomp $line;
    @data = split($delim, $line);

    # not-shared; ONDN; age>15;
    if( $data[$SharField] == 0 && $data[$AgeField] > 15
        && $data[$ModePrefField] != 6 && $data[$ModePrefField] != 0 ) {

        # print to new itdb
        print NEWITDB $line,"\n";
    }
}

############################

close ORIGITDB;
close NEWITDB;

############################
exit;
```

6.2.5 AddParkingCost.pl

```
#!/usr/bin/perl

# Parking Costs:
#
# Taz           Short   Long    Location
# -----         -----   ----    --------
# 1,2,10-16     2.47    4.94    CBD South of Burnside
# 3-6           1.59    3.18    CBD North of Burnside
# 43            1.38    2.76    Oregon Health Sciences University
# 510,934-936   0.80    1.59    Oregon City
# 846-847       0.00    3.03    Lloyd District
# 971-981       0.85    1.70    Vancouver WA

@ParkingCostShort = (0, 2.47, 1.59, 1.38, 0.80, 0.00, 0.85);
@ParkingCostLong  = (0, 4.94, 3.18, 2.76, 1.59, 3.03, 1.70);

open(FILE2, $ARGV[0]) or die "Can't open file $ARGV[0].\n";

while(<FILE2>) {
    $line = $_; chomp $line;
    @data = split(" ", $line);

    print $line,"\t",$ParkingCostLong[$data[4]],"\n";

}

close FILE2;
```

6.2.6 CombineDat.pl

```
#!/usr/bin/perl

$info = 1;

# Parking Costs:
#
# Taz           Short   Long    Location
# -----         -----   ----    --------
# 1,2,10-16     2.47    4.94    CBD South of Burnside
# 3-6           1.59    3.18    CBD North of Burnside
# 43            1.38    2.76    Oregon Health Sciences University
# 510,934-936   0.80    1.59    Oregon City
# 846-847       0.00    3.03    Lloyd District
# 971-981       0.85    1.70    Vancouver WA

@ParkingCostShort = (0, 2.47, 1.59, 1.38, 0.80, 0.00, 0.85);
@ParkingCostLong  = (0, 4.94, 3.18, 2.76, 1.59, 3.03, 1.70);

open(FILE1,$ARGV[0]) or die "Can't open file: $ARGV[0].\n";
for($i=0;$i<$info;$i++) {$line = <FILE1>;chomp $line; print $line;}

while(<FILE1>) {
    # read one line into scalars
    $line = $_; chomp $line;
    @data = split("\t", $line);
    # save in a hash
    $Rec0{$data[1]} = $line;
}
close FILE1;

open(FILE2, $ARGV[1]) or die "Can't open file $ARGV[1].\n";
for($i=0;$i<$info;$i++) {$line = <FILE2>;}
print "\tA-Time\tA-Time2\tA-Time3\tA-AnchorMod\tA-WalkOnly\tParkingCost\n";

while(<FILE2>) {
    $line = $_; chomp $line;
    @data = split(" ", $line);

    if( $Rec0{$data[1]} ) {
        print $Rec0{$data[1]},"\t",$data[5],"\t",$data[6],"\t",$data[7],"\t",
        $data[8],"\t",$data[9],"\t",$ParkingCostLong[$data[4]],"\n";
    }

}

close FILE2;
```

6.2.7 ItdbByTour.pl

```
#! /usr/bin/perl -w

#########################################################################

# This script takes an iteration database with the mandatory fields as
# specified below, and creates a new tour-based iteration database.

# The tour database will have four types of fields: traveler/household
# demographics, properties of the tour, properties of the home
# location, and properties of the primary anchor.

#########################################################################

# To get fields from itdb using gawk:
# head -2 itdb.000.it | tail -1 | gawk '{for(i=1;i<=NF;i++)print i-1,$i}' FS=","

$delim = ",";
$heads = 2;

# get command line arguments: original and new iteration database filenames
if($#ARGV+1 != 2) {
    print "\nusage: ItdbByTour.pl ORIGITDB TOURITDB\n\n";
    exit;
}
local($origitdb, $touritdb) = @ARGV;

# open input file
open(ORIGITDB, "$origitdb") || die "Failed to open $origitdb.";
# read header lines
for($i=0;$i<$heads;$i++) {
    $line = <ORIGITDB>;
}
# find fields automatically using header
chomp $line;
@data = split($delim, $line);
$i=0;
$HHField = $TravField = $TourField = $SubTField = $Act1Field = $Act2Field = $IncomeField
= $AgeField = $ActLoc1Field = $ActLoc2Field = $TypeField = $ModeField = $SharField =
$DurLBField = $DurUBField = $DriverField = $ParkingField = $Urban1Field = $Urban2Field =
$River1Field = $River2Field = $Tran1Field = $Tran2Field = $Zone1Field = $Zone2Field =
$TimeField = $DistField = $ModeStringField = -1;
while($data[$i]) {
    if($data[$i] eq "HH") {
        $HHField = $i;
    }
    elsif($data[$i] eq "TRAV") {
        $TravField = $i;
    }
    elsif($data[$i] eq "TOUR") {
        $TourField = $i;
    }
    elsif($data[$i] eq "SUBTOUR") {
        $SubTField = $i;
    }
    elsif($data[$i] eq "START_ACT_ID") {
        $Act1Field = $i;
    }
    elsif($data[$i] eq "END_ACT_ID") {
        $Act2Field = $i;
    }
    elsif($data[$i] eq "RHHINC") {
        $IncomeField = $i;
    }
    elsif($data[$i] eq "AGE") {
        $AgeField = $i;
    }
```

```
    elsif($data[$i] eq "START_ACT_LOCATION") {
        $ActLoc1Field = $i;
    }
    elsif($data[$i] eq "END_ACT_LOCATION") {
        $ActLoc2Field = $i;
    }
    elsif($data[$i] eq "END_ACT_TYPE") {
        $TypeField = $i;
    }
    elsif($data[$i] eq "END_MODE_PREF") {
        $ModeField = $i;
    }
    elsif($data[$i] eq "END_OTHER_PARTICIPANTS") {
        $SharField = $i;
    }
    elsif($data[$i] eq "END_DUR_UB") {
        $DurUBField = $i;
    }
    elsif($data[$i] eq "END_DUR_LB") {
        $DurLBField = $i;
    }
    elsif($data[$i] eq "DRIVER_ID") {
        $DriverField = $i;
    }
    elsif($data[$i] eq "END_ACT_USER_DATA_PARKING_ZN") { # *
        $ParkingField = $i;
    }
    elsif($data[$i] eq "START_ACT_USER_DATA_URBAN_TYPE") { # *
        $Urban1Field = $i;
    }
    elsif($data[$i] eq "END_ACT_USER_DATA_URBAN_TYPE") { # *
        $Urban2Field = $i;
    }
    elsif($data[$i] eq "START_ACT_USER_DATA_River_Zone") { # *
        $River1Field = $i;
    }
    elsif($data[$i] eq "END_ACT_USER_DATA_River_Zone") { # *
        $River2Field = $i;
    }
    elsif($data[$i] eq "START_ACT_USER_DATA_Tran_Dist") {
        $Tran1Field = $i;
    }
    elsif($data[$i] eq "END_ACT_USER_DATA_Tran_Dist") {
        $Tran2Field = $i;
    }
    elsif($data[$i] eq "START_ACT_USER_DATA_Zones_8") {
        $Zone1Field = $i;
    }
    elsif($data[$i] eq "END_ACT_USER_DATA_Zones_8") {
        $Zone2Field = $i;
    }
    elsif($data[$i] eq "DURATION") {
    #elsif($data[$i] eq "TIME_SUM") {
        $TimeField = $i;
    }
    elsif($data[$i] eq "EUCLID") {
    #elsif($data[$i] eq "DISTANCE_SUM") {
        $DistField = $i;
    }
    elsif($data[$i] eq "MODE_STRING") {
        $ModeStringField = $i;
    }
    $i++;
}
# check for mandatory fields
$Missing = 0;
if($HHField == -1) {
    print "Missing HH.\n";
    $Missing++;
}
```

```
if($TravField == -1) {
    print "Missing TRAV.\n";
    $Missing++;
}
if($TourField == -1) {
    print "Missing TOUR.\n";
    $Missing++;
}
if($SubTField == -1) {
    print "Missing SUBTOUR.\n";
    $Missing++;
}
if($Act1Field == -1) {
    print "Missing START_ACT_ID.\n";
    $Missing++;
}
if($Act2Field == -1) {
    print "Missing END_ACT_ID.\n";
    $Missing++;
}
if($IncomeField == -1) {
    print "Missing RHHINC.\n";
    $Missing++;
}
if($AgeField == -1) {
    print "Missing AGE.\n";
    $Missing++;
}
if($ActLoc1Field == -1) {
    print "Missing START_ACT_LOCATION.\n";
    $Missing++;
}
if($ActLoc2Field == -1) {
    print "Missing END_ACT_LOCATION.\n";
    $Missing++;
}
if($TypeField == -1) {
    print "Missing END_ACT_TYPE.\n";
    $Missing++;
}
if($ModeField == -1) {
    print "Missing END_MODE_PREF.\n";
    $Missing++;
}
if($SharField == -1) {
    print "Missing END_OTHER_PARTICIPANTS.\n";
    $Missing++;
}
if($DurUBField == -1) {
    print "Missing END_DUR_UB.\n";
    $Missing++;
}
if($DurLBField == -1) {
    print "Missing END_DUR_LB.\n";
    $Missing++;
}
if($DriverField == -1) {
    print "Missing DRIVER_ID.\n";
    $Missing++;
}
if($ParkingField == -1) {
    print "Missing END_ACT_USER_DATA_PARKING_ZN.\n";
    $Missing++;
}
if($Urban1Field == -1) {
    print "Missing START_ACT_USER_DATA_URBAN_TYPE.\n";
    $Missing++;
}
```

```
if($Urban2Field == -1) {
    print "Missing END_ACT_USER_DATA_URBAN_TYPE.\n";
    $Missing++;
}
if($River1Field == -1) {
    print "Missing START_ACT_USER_DATA_River_Zone.\n";
    $Missing++;
}
if($River2Field == -1) {
    print "Missing END_ACT_USER_DATA_River_Zone.\n";
    $Missing++;
}
if($Tran1Field == -1) {
    print "Missing START_ACT_USER_DATA_Tran_Dist.\n";
    $Missing++;
}
if($Tran2Field == -1) {
    print "Missing END_ACT_USER_DATA_Tran_Dist.\n";
    $Missing++;
}
if($Zone1Field == -1) {
    print "Missing START_ACT_USER_DATA_Zones_8.\n";
    $Missing++;
}
if($Zone2Field == -1) {
    print "Missing END_ACT_USER_DATA_Zones_8.\n";
    $Missing++;
}
if($TimeField == -1) {
    print "Missing DURATION.\n";
    $Missing++;
}
if($DistField == -1) {
    print "Missing EUCLID.\n";
    $Missing++;
}
if($ModeStringField == -1) {
    print "Missing MODE_STRING.\n";
    $Missing++;
}
if($Missing > 0) {exit;}

# open output files, print new header
open(TOURITDB, ">$touritdb") || die "Failed to open $touritdb.";
print TOURITDB "HH\tTrav\tIncome\tAge\t";
print TOURITDB
"Tour\tTourAct\tType\tMode\tModePref\tTime\tDistance\tMulitMode\tShared\tSubTours";
print TOURITDB "\tCrossColumbia\tCrossWillamette\tRailOnly\tMaxTranDist\tModeString\t";
print TOURITDB "HomeZone\tHomeRiver\tHomeUrban\tHomeTranDist\t";
print TOURITDB
"AnchorAct\tAnchorZone\tAnchorRiver\tAnchorUrban\tAnchorParking\tAnchorTranDist\t";
print TOURITDB "AnchorModPref\tAnchorMod\tAnchorType\tDickTime2\tDickTime3\n";

##########################################################

$Tour = -1;
$NotFirstTrav = 0;
$Time = "NA";

# loop over lines
while ( <ORIGITDB> ) {
    $line = $_; chomp $line;
    @data = split($delim, $line);
    if($data[$Act1Field] != $data[$Act2Field]) { # skip "activity" entries
        if($Tour == $data[$TourField] && $Trav == $data[$TravField]) { # same tour
            if($data[$ModeField] != $CurrentMode) { $MultiMode = 1; }
            if($data[$ModeField] == 2 && $data[$River1Field] != $data[$River2Field]) {
                if($data[$River1Field] == 1 || $data[$River2Field] == 1) { $River1++; }
                else { $River2++; }
            }
```

```
            $CurrentMode = $data[$ModeField];
            $LastLoc = $data[$ActLoc2Field];
            if($data[$ModeField] == 4) { $RailOnly++; }
            if($MaxTranDist < $data[$Tran2Field]) { $MaxTranDist = $data[$Tran2Field]; }
            if($data[$SubTField] == 0) { # ignore sub-tours
                $GoodTour = ($data[$TimeField] ne "NA") ? $GoodTour : 0; # check Router field
                if($GoodTour == 1) { $Time += $data[$TimeField]; } # can't accumulate "NA"
                $Dist += $data[$DistField];
                $ModeString .= $data[$ModeStringField];
                $ActMode .= $data[$ModeField];
                if($data[$SharField] != 0) {
                    if($data[$DriverField] == $Trav) { $Shared = 4; } # main-tour shared-
ride driver
                    elsif($Shared < 2) {$Shared = 2;} # main-tour shared-ride passenger
                }
                if($LastLoc != $HomeLoc) { # final at-home activity cannot be anchor
                    $ActivDuration = ($data[$DurLBField] + $data[$DurUBField])/2;
                    $NewAnchor = 0;
                    # work-school--or-college anchor:
                    if($data[$TypeField] == 1 || $data[$TypeField] == 7 ||
$data[$TypeField] == 8) {
                        if($TourType ne "work" || $ActivDuration > $AnchorDuration) {
                            $NewAnchor = 1; # better anchor type, or better work anchor
                            $TourType = "work";
                        }
                    }
                    elsif($TourType ne "work" && $data[$TypeField] == 2) { # shop anchor
                        if($TourType ne "shop" || $ActivDuration > $AnchorDuration) {
                            $NewAnchor = 1; # better anchor type, or better shop anchor
                            $TourType = "shop";
                        }
                     }
                     elsif($TourType ne "work" && $TourType ne "shop"
                          && $ActivDuration > $AnchorDuration) { # "other" type anchor, default
                        $NewAnchor = 1; # better anchor
                    } # note precedence of w, s, then o anchor locations
                    if($NewAnchor == 1) {
                        # get new anchor data
                        $AnchorAct = $data[$Act2Field];
                        $AnchorZone = $data[$Zone2Field];
                        $AnchorRiver = $data[$River2Field];
                        $AnchorUrban = $data[$Urban2Field];
                        $AnchorParking = $data[$ParkingField];
                        $AnchorTranDist = $data[$Tran2Field];
                        $AnchorDuration = $ActivDuration;
                        $AnchorModePref = $CurrentMode;
                        $AnchorMode = $ModeString;
                        $AnchorType = $data[$TypeField];
                        $DickTime2 = $Time;
                        $DickTime3 = $data[$TimeField];
                    }
                } # ignoreA at-home activities
            } # not a sub-tour
            else { # is a subtour
                    if($data[$SharField] != 0) {
                    if($data[$DriverField] == $Trav && $Shared < 3) { $Shared = 3; } #
sub-tour shared-ride driver
                    elsif($Shared == 0) {$Shared = 1;} # sub-tour shared-ride passenger
                }
                # count
                $Nsubtours++;
            }
        } # same tour
        else { # new tour or HH
            if($NotFirstTrav == 1) {
                # first determine tour-mode
                $TmpModeString = $ModeString;
                # test for illegal modes
                if($TmpModeString =~ /U/ || $TmpModeString =~ /NA/ || $TmpModeString =~
/p/ || $TmpModeString =~ /t/) {
```

```
                    $TourMode = -1; # "bad mode"
                    }
                else {
                    # then get rid of magic k & K:
                    $TmpModeString =~ s/K/k/g;
                    # ignore bike for now:
                    $TmpModeString =~ s/i/k/g;
                    # determine mode for tour
                    if($TmpModeString =~ /c/) {
                        if($TmpModeString =~ /l/ || $TmpModeString =~ /b/) {
                            $TourMode = 6; # "possible P&R mode"
                    }
                        else {
                            $TourMode = 2; # "car-only mode"
                        }
                    }
                    elsif($TmpModeString =~ /b/) {
                        if($TmpModeString =~ /l/) {
                            $TourMode = 5; # "mixed-transit mode"
                        }
                        else {
                            $TourMode = 3; # "bus-only mode"
                        }
                    }
                    elsif($TmpModeString =~ /l/) {
                        $TourMode = 4; # "rail-only mode"
                    }
                    elsif($TmpModeString =~ /w/) {
                       $TourMode = 1; # "walk-only mode"
                    }
                    elsif($TmpModeString =~ /k/) {
                       $TourMode = 0; # "magic-only mode (includes k,K,i)"
                    }
                }

                # determine primary anchor mode
                $TmpModeString = $AnchorMode;
                # test for illegal modes
                if($TmpModeString =~ /U/ || $TmpModeString =~ /NA/ || $TmpModeString =~
/p/ || $TmpModeString =~ /t/) {
                    $AnchorMode = -1; # "bad mode"
                }
                else {
                    # then get rid of magic k & K:
                    $TmpModeString =~ s/K/k/g;
                    # ignore bike for now:
                    $TmpModeString =~ s/i/k/g;
                    # determine mode for tour
                    if($TmpModeString =~ /c/) {
                        if($TmpModeString =~ /l/ || $TmpModeString =~ /b/) {
                            $AnchorMode = 6; # "possible P&R mode"
                        }
                        else {
                            $AnchorMode = 2; # "car-only mode"
                        }
                    }
                    elsif($TmpModeString =~ /b/) {
                        if($TmpModeString =~ /l/) {
                            $AnchorMode = 5; # "mixed-transit mode"
                        }
                        else {
                            $AnchorMode = 3; # "bus-only mode"
                        }
                    }
                    elsif($TmpModeString =~ /l/) {
                        $AnchorMode = 4; # "rail-only mode"
                    }
                    elsif($TmpModeString =~ /w/) {
                        $AnchorMode = 1; # "walk-only mode"
                    }
```

```
                    elsif($TmpModeString =~ /k/) {
                        $AnchorMode = 0; # "magic-only mode (includes k,K,i)"
                    }
                }

                #determine tour mode preference (activity modes)
                $TmpModeString = $ActMode;
                # test for illegal modes
                if($TmpModeString =~ /NA/) {
                    $TourModePref = -1; # "bad mode"
                }
                else {
                    # then get rid of magic k & K:
                    $TmpModeString =~ s/8/0/g;
                    $TmpModeString =~ s/9/0/g;
                    # ignore bike for now:
                    $TmpModeString =~ s/7/0/g;
                    # determine mode for tour
                    if($TmpModeString =~ /2/) {
                        if($TmpModeString =~ /6/ || $TmpModeString =~ /5/
                           || $TmpModeString =~ /4/ || $TmpModeString =~ /3/) {
                            $TourModePref = 6; # "possible P&R mode"
                        }
                        else {
                            $TourModePref = 2; # "car-only mode"
                        }
                    }
                    elsif($TmpModeString =~ /3/) {
                        if($TmpModeString =~ /4/) {
                            $TourModePref = 5; # "mixed-transit mode"
                        }
                        else {
                            $TourModePref = 3; # "bus-only mode"
                        }
                    }
                    elsif($TmpModeString =~ /4/) {
                        $TourModePref = 4; # "rail-only mode"
                    }
                    elsif($TmpModeString =~ /1/) {
                        $TourModePref = 1; # "walk-only mode"
                    }
                    elsif($TmpModeString =~ /0/) {
                        $TourModePref = 0; # "magic-only mode (includes k,K,i)"
                    }
                }


                # print info from last tour
                print TOURITDB $HH,"\t",$Trav,"\t",$Income,"\t",$Age,"\t";
                print TOURITDB
$Tour,"\t",$TourAct,"\t",$TourType,"\t",$TourMode,"\t",$TourModePref,"\t";
                if($GoodTour == 1 && $LastLoc == $HomeLoc) { # only OK if they had times
& got back home
                    print TOURITDB $Time;
                }
                else {
                    print TOURITDB "NA";
                }
                print TOURITDB
"\t",$Dist,"\t",$MultiMode,"\t",$Shared,"\t",$Nsubtours,"\t";
                print TOURITDB
$River1,"\t",$River2,"\t",$RailOnly,"\t",$MaxTranDist,"\t",$ModeString,"\t";
                print TOURITDB
$HomeZone,"\t",$HomeRiver,"\t",$HomeUrban,"\t",$HomeTranDist,"\t";
                print TOURITDB
$AnchorAct,"\t",$AnchorZone,"\t",$AnchorRiver,"\t",$AnchorUrban,"\t";
                print TOURITDB
$AnchorParking,"\t",$AnchorTranDist,"\t",$AnchorModePref,"\t",$AnchorMode,"\t";
                print TOURITDB $AnchorType,"\t",$DickTime2,"\t",$DickTime3,"\n";
            }
            $NotFirstTrav = 1;
```

```
            # Traveler data
            $HH     = $data[$HHField];
            $Trav   = $data[$TravField];
            $Income = $data[$IncomeField];
            $Age    = $data[$AgeField];
            # Tour data
            $Tour    = $data[$TourField];
            $TourAct = $data[$Act2Field];
            $GoodTour = ($data[$TimeField] ne "NA") ? 1 : 0;
            if($GoodTour == 1) { $Time = $data[$TimeField]; }
            $Dist = $data[$DistField];
            $ModeString = $data[$ModeStringField];
            $ActMode = $data[$ModeField];
            if($data[$SharField] != 0) {
                if($data[$DriverField] == $Trav) { $Shared = 4; } # main-tour shared-ride driver
                else {$Shared = 2;} # main-tour shared-ride passenger
            }
            else {$Shared = 0;}
            $CurrentMode = $data[$ModeField];
            $MultiMode = 0;
            $AnchorDuration = 0;
            $Nsubtours = 0;
            if($data[$ModeField] == 2 && $data[$River1Field] != $data[$River2Field]) {
                if($data[$River1Field] == 1 || $data[$River2Field] == 1) { $River1 = 1;
$River2 = 0; }
            else { $River1 = 0; $River2 = 1; }
            }
            else { $River1 = 0; $River2 = 0; }
            if($data[$ModeField] == 4) { $RailOnly = 1; }
            else { $RailOnly = 0; }
            $MaxTranDist = ($data[$Tran1Field] > $data[$Tran2Field]) ? $data[$Tran1Field]
: $data[$Tran2Field];
            # Home data
            $HomeLoc = $data[$ActLoc1Field];
            $HomeZone = $data[$Zone1Field];
            $HomeRiver = $data[$River1Field];
            $HomeUrban = $data[$Urban1Field];
            $HomeTranDist = $data[$Tran1Field];
            # Anchor data
            $AnchorAct = $data[$Act2Field];
            $AnchorZone = $data[$Zone2Field];
            $AnchorRiver = $data[$River2Field];
            $AnchorUrban = $data[$Urban2Field];
            $AnchorParking = $data[$ParkingField];
            $AnchorTranDist = $data[$Tran2Field];
            $AnchorDuration = ($data[$DurLBField] + $data[$DurUBField])/2;
            $AnchorModePref = $CurrentMode;
            $AnchorMode = $ModeString;
            $AnchorType = $data[$TypeField];
            $DickTime2 = $Time;
            $DickTime3 = $data[$TimeField];
            # work-school-or-college anchor:
           if($data[$TypeField] == 1 || $data[$TypeField] == 7 || $data[$TypeField] == 8) {
                $TourType = "work";
            }
            elsif($data[$TypeField] == 2) { # shop anchor
                $TourType = "shop";
            }
            else { # "other" type anchor, default
                $TourType = "other";
            } # note precedence of w, s, then o anchor locations
        } # new tour
    } # skip "activity" entries
}

#########################################################
```

```
# first determine tour-mode

$TmpModeString = $ModeString;
# test for illegal modes
if($TmpModeString =~ /U/ || $TmpModeString =~ /NA/ || $TmpModeString =~ /p/ ||
$TmpModeString =~ /t/) {
    $TourMode = -1; # "bad mode"
}
else {
    # then get rid of magic k & K:
    $TmpModeString =~ s/K/k/g;
    # ignore bike for now:
    $TmpModeString =~ s/i/k/g;
    # determine mode for tour
    if($TmpModeString =~ /c/) {
        if($TmpModeString =~ /l/ || $TmpModeString =~ /b/) {
            $TourMode = 6; # "possible P&R mode"
        }
        else {
            $TourMode = 2; # "car-only mode"
        }
    }
    elsif($TmpModeString =~ /b/) {
        if($TmpModeString =~ /l/) {
            $TourMode = 5; # "mixed-transit mode"
        }
        else {
            $TourMode = 3; # "bus-only mode"
        }
    }
    elsif($TmpModeString =~ /l/) {
        $TourMode = 4; # "rail-only mode"
    }
    elsif($TmpModeString =~ /w/) {
        $TourMode = 1; # "walk-only mode"
    }
    elsif($TmpModeString =~ /k/) {
        $TourMode = 0; # "magic-only mode (includes k,K,i)"
    }
}

# determine primary anchor mode
$TmpModeString = $AnchorMode;
# test for illegal modes
if($TmpModeString =~ /U/ || $TmpModeString =~ /NA/ || $TmpModeString =~ /p/ ||
$TmpModeString =~ /t/) {
    $AnchorMode = -1; # "bad mode"
}
else {
    # then get rid of magic k & K:
    $TmpModeString =~ s/K/k/g;
    # ignore bike for now:
    $TmpModeString =~ s/i/k/g;
    # determine mode for tour
    if($TmpModeString =~ /c/) {
        if($TmpModeString =~ /l/ || $TmpModeString =~ /b/) {
            $AnchorMode = 6; # "possible P&R mode"
        }
        else {
            $AnchorMode = 2; # "car-only mode"
        }
    }
    elsif($TmpModeString =~ /b/) {
        if($TmpModeString =~ /l/) {
            $AnchorMode = 5; # "mixed-transit mode"
        }
        else {
            $AnchorMode = 3; # "bus-only mode"
       }
    }
```

```
    elsif($TmpModeString =~ /l/) {
        $AnchorMode = 4; # "rail-only mode"
    }
    elsif($TmpModeString =~ /w/) {
        $AnchorMode = 1; # "walk-only mode"
    }
    elsif($TmpModeString =~ /k/) {
        $AnchorMode = 0; # "magic-only mode (includes k,K,i)"
    }
}

#determine tour mode preference (activity modes)
$TmpModeString = $ActMode;
# test for illegal modes
if($TmpModeString =~ /NA/) {
    $TourModePref = -1; # "bad mode"
}
else {
    # then get rid of magic k & K:
    $TmpModeString =~ s/8/0/g;
    $TmpModeString =~ s/9/0/g;
    # ignore bike for now:
    $TmpModeString =~ s/7/0/g;
    # determine mode for tour
    if($TmpModeString =~ /2/) {
        if($TmpModeString =~ /6/ || $TmpModeString =~ /5/
           || $TmpModeString =~ /4/ || $TmpModeString =~ /3/) {
            $TourModePref = 6; # "possible P&R mode"
        }
        else {
            $TourModePref = 2; # "car-only mode"
        }
    }
    elsif($TmpModeString =~ /3/) {
        if($TmpModeString =~ /4/) {
            $TourModePref = 5; # "mixed-transit mode"
        }
        else {
            $TourModePref = 3; # "bus-only mode"
        }
    }
    elsif($TmpModeString =~ /4/) {
   $TourModePref = 4; # "rail-only mode"
    }
    elsif($TmpModeString =~ /1/) {
        $TourModePref = 1; # "walk-only mode"
    }
    elsif($TmpModeString =~ /0/) {
        $TourModePref = 0; # "magic-only mode (includes k,K,i)"
    }
}

# print info from last tour
print TOURITDB $HH,"\t",$Trav,"\t",$Income,"\t",$Age,"\t";
print TOURITDB $Tour,"\t",$TourAct,"\t",$TourType,"\t",$TourMode,"\t",$TourModePref,"\t";
if($GoodTour == 1 && $LastLoc == $HomeLoc) { # only OK if they had times & got back home
    print TOURITDB $Time;
}
else {
    print TOURITDB "NA";
}
print TOURITDB "\t",$Dist,"\t",$MultiMode,"\t",$Shared,"\t",$Nsubtours,"\t";
print TOURITDB
$River1,"\t",$River2,"\t",$RailOnly,"\t",$MaxTranDist,"\t",$ModeString,"\t";
print TOURITDB $HomeZone,"\t",$HomeRiver,"\t",$HomeUrban,"\t",$HomeTranDist,"\t";
print TOURITDB $AnchorAct,"\t",$AnchorZone,"\t",$AnchorRiver,"\t",$AnchorUrban,"\t";
print TOURITDB
$AnchorParking,"\t",$AnchorTranDist,"\t",$AnchorModePref,"\t",$AnchorMode,"\t";
print TOURITDB $AnchorType,"\t",$DickTime2,"\t",$DickTime3,"\n";
```

```
close ORIGITDB;
close TOURITDB;

exit
```

6.2.8 ReModeTrans.pl

```
#! /usr/bin/perl -w

##########################################################################

# NOTE: This takes a TOUR-based itdb as input

# This script creates feedback commands that will convert ALL rail or
# walk trips in the tours in a tour database into regular transit
# trips.

# - create activity feedback file with re-mode commands
# - create a router household file for router feedback

##########################################################################

$delim = "\t";
$heads = 1;

# get command line arguments:
if($#ARGV+1 != 3) {
    print "\nusage: ReModeTrans.pl TOURITDB ROUTEFB ACTIVFB\n\n";
    print "   TOURITDB = input tour-based, tab delimited iteration database\n";
    print "   ROUTEFB  = household file changing w,l-trips into t-trips\n";
    print "   ACTIVFB  = activity regenerator feedback command file\n\n";
    exit;
}
local($origitdb, $routefile, $activfile) = @ARGV;

# open input itdb and parse header
open(ORIGITDB, "$origitdb") || die "Failed to open $origitdb.";
# read header lines
for($i=0;$i<$heads;$i++) {
    $line = <ORIGITDB>;
}

# find fields automatically using header
chomp $line;
@data = split($delim, $line);
$i=0;
$HHField = $ActField = -1;
while($data[$i]) {
    if($data[$i] eq "HH") {
        $HHField = $i;
    }
    elsif($data[$i] eq "AnchorAct") {
        $ActField = $i;
    }
    $i++;
}
# check for mandatory fields
$Missing = 0;
if($HHField == -1) {
    print "Missing HH.\n";
    $Missing++;
}
if($ActField == -1) {
    print "Missing AnchorAct.\n";
    $Missing++;
}
if($Missing > 0) {exit;}


# open output files
open(ROUTEFB, ">$routefile") || die "Failed to open $routefile.";
open(ACTIVFB, ">$activfile") || die "Failed to open $activfile.";
```

```
############################

$HH2 = -999;

# loop over lines
while ( <ORIGITDB> ) {
    $line = $_; chomp $line;
    @data = split($delim, $line);

    $HH = $data[$HHField];

    # send mode-change feedback command to activity feedback file
    print ACTIVFB $HH," ",$data[$ActField]," M 3 1 2 4\n";
    # send HH to router feedback file if not already sent
    if($HH2 != $HH) {
        $HH2 = $HH;
        print ROUTEFB $HH,"\n";
    }
}

############################

close ORIGITDB;
close ROUTEFB;
close ACTIVFB;

############################
exit
```

6.2.9 Select-Random.pl

```
#! /usr/bin/perl -w

# get command line arguments
local($seed, $alpha, $infile, $outfile, $outfile2) = @ARGV;

# set random number seed
srand($seed);

# open input and output files
open(REPLANNED, "$infile") || print "Failed to open $infile.";
open(SELECTED, ">$outfile") || print "Failed to open $outfile.";
open(SELECTED2, ">$outfile2") || print "Failed to open $outfile2.";

# output any line for which the random number is less than $alpha
while ( <REPLANNED> ) {
    $random = rand();
    #print STDOUT $random, "\n";
    if ( $random < $alpha ) {
        print SELECTED;
    }
    else{
        print SELECTED2;
    }
}
```

6.2.10 Match_Pop.pl

```
#!/usr/bin/perl

$info = 2;

open(FILE1,$ARGV[0]) or die "Can't open file: $ARGV[0].\n";
while(<FILE1>) {
    # read one line into scalars
    $line = $_; chop $line;
    @data = split("\t", $line);
    # save in a hash
    $Rec0{$data[0]} = 1;
    $Rec1{$data[1]} = 1;
}
close FILE1;

open(FILE2, $ARGV[1]) or die "Can't open file $ARGV[1].\n";
open(FILE3, ">$ARGV[2]") or die "Can't open file $ARGV[2].\n";
for($i=0;$i<$info;$i++) {$line = <FILE2>;print FILE3 $line;}

while(<FILE2>) {
    $line = $_; chop $line;
    @data = split(" ", $line);

    if( $data[2] eq "H" && $Rec0{$data[3]} ) {
       $data[4] = 1;
       print FILE3 join(" ",@data),"\n";
    }
    elsif( $data[1] eq "P" && $Rec1{$data[2]} )  {
         print FILE3 $line,"\n";
    }
}

close FILE2;
close FILE3;
```

6.2.11 Match_Acts.pl

```
#!/usr/bin/perl
# Solaris: /sw/Cvol/bin/perl

# NOTE: indexing starts at ZERO in perl!

# File1 is the household file, File2 is the input activity file, File3
# is the output activity file with matches in the HH file, and File 4
# is the output activity file for households not in the HH file.

$delim = "\t";
#$delim = " ";
$info = 0;

open(FILE1,$ARGV[0]) or die "Can't open file: $ARGV[0].\n";
while(<FILE1>) {
    # read one line into scalars
    $line = $_; chop $line;
    @data = split($delim, $line);
    # save in a hash
    $Rec1{$data[1]} = 1;
    $Rec4{$data[1]} = $data[4];
}
close FILE1;

open(FILE2, $ARGV[1]) or die "Can't open file $ARGV[1].\n";
open(FILE3, ">$ARGV[2]") or die "Can't open file $ARGV[2].\n";
for($i=0;$i<$info;$i++) {$line = <FILE2>;if($i==$info-1){print FILE3 $line;}}

$LastTrav = -1;

while(<FILE2>) {
    $line = $_; chop $line;
    @data = split($delim, $line);

    if( $Rec1{$data[1]} ) {
        $Trav = $data[1];

        if( $LastTrav != $Trav ) {
            $Home = $data[20];
            $TourNum = 1;
        }
        elsif( $Home == $data[20] ) {
            if( $LastLoc != $Home ) {  # account for Stephen's Collator
                $TourNum++;
            }
        }

        if( $TourNum == $Rec4{$data[1]}
            || ($TourNum-1 == $Rec4{$data[1]} && $Home == $data[20])) {
            print FILE3 $line,"\n";
        }

        $LastTrav = $Trav;
        $LastLoc = $data[20];
    }
}

close FILE2;
close FILE3;
```

6.2.12 Stratify.pl

```
#! /usr/bin/perl -w

##########################################################################

# This script takes any tour-based database and creates the 18
# separate files corresponding to the 18 strata decided upon by METRO:
#   1) work / non-work
#   2) home urbanization: (very urban, urban, sub-urban) = (1,2,3)
#   3) primary anchor urbanization: very urban, urban, sub-urban

# If no 2nd argument is given, then no output is made to files, but
# the counts-by-bin are sent to standard output.

# NOTE: This script will work independent of optional fields.

##########################################################################

$heads = 1;
$delim = "\t";
$Output = 0;

# get command line arguments:
if($#ARGV+1 == 1) {
    local($origitdb) = @ARGV;
}
elsif($#ARGV+1 == 2) {
    local($origitdb, $outitdb) = @ARGV;
    $Output = 1;
}
else {
    print "\nusage: StratifyDB.pl TOURITDB [ NEWDBPREFIX ]\n\n";
    print "   TOURITDB = input tour-based, tab delimited iteration database\n";
    print "   NEWDBPREFIX = output as TOURITDB, but only selected tours\n";
    print "                 in each of 18 files - one for each stratum.\n\n";
    exit;
}

local($origitdb, $outitdb) = @ARGV;

# open input itdb and parse header
open(ORIGITDB, "$origitdb") || die "Failed to open $origitdb.";
# read header lines
for($i=0;$i<$heads;$i++) {
    $line = <ORIGITDB>;
}

# find fields automatically using header
chomp $line;
@data = split($delim, $line);
$i=0;
$TypeField = $Urban1Field = $Urban2Field = -1;
while($data[$i]) {
    if($data[$i] eq "Type") {
        $TypeField = $i;
    }
    elsif($data[$i] eq "HomeUrban") {
        $Urban1Field = $i;
    }
    elsif($data[$i] eq "AnchorUrban") {
        $Urban2Field = $i;
    }
    $i++;
}
# check for mandatory fields
$Missing = 0;
if($TypeField == -1) {
```

```
    print "Missing Type.\n";
    $Missing++;
}
if($Urban1Field == -1) {
    print "Missing HomeUrban.\n";
    $Missing++;
}
if($Urban2Field == -1) {
    print "Missing AnchorUrban.\n";
    $Missing++;
}
if($Missing > 0) {exit;}


#########################################################

if($Output == 1) { # stratify DB into files

    # open output files - brute force for now
    open(NEWITDBw11, ">$outitdb.w.1.1") || die "Failed to open $outitdb.w.1.1.";
    open(NEWITDBw12, ">$outitdb.w.1.2") || die "Failed to open $outitdb.w.1.2.";
    open(NEWITDBw13, ">$outitdb.w.1.3") || die "Failed to open $outitdb.w.1.3.";
    open(NEWITDBw21, ">$outitdb.w.2.1") || die "Failed to open $outitdb.w.2.1.";
    open(NEWITDBw22, ">$outitdb.w.2.2") || die "Failed to open $outitdb.w.2.2.";
    open(NEWITDBw23, ">$outitdb.w.2.3") || die "Failed to open $outitdb.w.2.3.";
    open(NEWITDBw31, ">$outitdb.w.3.1") || die "Failed to open $outitdb.w.3.1.";
    open(NEWITDBw32, ">$outitdb.w.3.2") || die "Failed to open $outitdb.w.3.2.";
    open(NEWITDBw33, ">$outitdb.w.3.3") || die "Failed to open $outitdb.w.3.3.";
    open(NEWITDBn11, ">$outitdb.n.1.1") || die "Failed to open $outitdb.n.1.1.";
    open(NEWITDBn12, ">$outitdb.n.1.2") || die "Failed to open $outitdb.n.1.2.";
    open(NEWITDBn13, ">$outitdb.n.1.3") || die "Failed to open $outitdb.n.1.3.";
    open(NEWITDBn21, ">$outitdb.n.2.1") || die "Failed to open $outitdb.n.2.1.";
    open(NEWITDBn22, ">$outitdb.n.2.2") || die "Failed to open $outitdb.n.2.2.";
    open(NEWITDBn23, ">$outitdb.n.2.3") || die "Failed to open $outitdb.n.2.3.";
    open(NEWITDBn31, ">$outitdb.n.3.1") || die "Failed to open $outitdb.n.3.1.";
    open(NEWITDBn32, ">$outitdb.n.3.2") || die "Failed to open $outitdb.n.3.2.";
    open(NEWITDBn33, ">$outitdb.n.3.3") || die "Failed to open $outitdb.n.3.3.";

    # print itdb header
    print NEWITDBw11 $line,"\n";
    print NEWITDBw12 $line,"\n";
    print NEWITDBw13 $line,"\n";
    print NEWITDBw21 $line,"\n";
    print NEWITDBw22 $line,"\n";
    print NEWITDBw23 $line,"\n";
    print NEWITDBw31 $line,"\n";
    print NEWITDBw32 $line,"\n";
    print NEWITDBw33 $line,"\n";
    print NEWITDBn11 $line,"\n";
    print NEWITDBn12 $line,"\n";
    print NEWITDBn13 $line,"\n";
    print NEWITDBn21 $line,"\n";
    print NEWITDBn22 $line,"\n";
    print NEWITDBn23 $line,"\n";
    print NEWITDBn31 $line,"\n";
    print NEWITDBn32 $line,"\n";
    print NEWITDBn33 $line,"\n";

    # loop over lines
    while ( <ORIGITDB> ) {
        $line = $_; chomp $line;
        @data = split($delim, $line);

        # send to appropriate file
        if($data[$TypeField] eq "work") {
            if($data[$Urban1Field] == 1) {
                if($data[$Urban2Field] == 1) {
                    print NEWITDBw11 $line,"\n";
                }
```

```
            elsif($data[$Urban2Field] == 2) {
                print NEWITDBw12 $line,"\n";
            }
            else {
                print NEWITDBw13 $line,"\n";
            }
        }
        elsif($data[$Urban1Field] == 2) {
            if($data[$Urban2Field] == 1) {
                print NEWITDBw21 $line,"\n";
            }
            elsif($data[$Urban2Field] == 2) {
                print NEWITDBw22 $line,"\n";
            }
            else {
                print NEWITDBw23 $line,"\n";
            }
        }
        else { # home urban = 3
            if($data[$Urban2Field] == 1) {
                print NEWITDBw31 $line,"\n";
            }
            elsif($data[$Urban2Field] == 2) {
                print NEWITDBw32 $line,"\n";
            }
            else {
                print NEWITDBw33 $line,"\n";
            }
        }
    }
    else { # non-work
        if($data[$Urban1Field] == 1) {
            if($data[$Urban2Field] == 1) {
                print NEWITDBn11 $line,"\n";
            }
            elsif($data[$Urban2Field] == 2) {
                print NEWITDBn12 $line,"\n";
            }
            else { # work urban = 3
                print NEWITDBn13 $line,"\n";
            }
        }
        elsif($data[$Urban1Field] == 2) {
            if($data[$Urban2Field] == 1) {
                print NEWITDBn21 $line,"\n";
            }
            elsif($data[$Urban2Field] == 2) {
                print NEWITDBn22 $line,"\n";
            }
            else {
                print NEWITDBn23 $line,"\n";
            }
        }
        else { # home urban = 3
            if($data[$Urban2Field] == 1) {
                print NEWITDBn31 $line,"\n";
            }
            elsif($data[$Urban2Field] == 2) {
                print NEWITDBn32 $line,"\n";
            }
            else { # work urban = 3
                print NEWITDBn33 $line,"\n";
            }
        }
    }
}

close NEWITDBw11;
close NEWITDBw12;
close NEWITDBw13;
close NEWITDBw21;
```

```
    close NEWITDBw22;
    close NEWITDBw23;
    close NEWITDBw31;
    close NEWITDBw32;
    close NEWITDBw33;
    close NEWITDBn11;
    close NEWITDBn12;
    close NEWITDBn13;
    close NEWITDBn21;
    close NEWITDBn22;
    close NEWITDBn23;
    close NEWITDBn31;
    close NEWITDBn32;
    close NEWITDBn33;
}

#########################################################

if($Output == 0) { # no output files

    # loop over lines
    while ( <ORIGITDB> ) {
        $line = $_; chomp $line;
        @data = split($delim, $line);

        # count by strata
        if($data[$TypeField] eq "work") { $work = "w"; }
        else { $work = "n"; }
        $Counts{$work}{$data[$Urban1Field]}{$data[$Urban2Field]}++;
    }

    # print results
    print "Work/N\tHomUrb\tAnchUrb\tCount\n";
    foreach $key1 (sort keys %Counts) {
        foreach $key2 (sort keys %{ $Counts{$key1} } ) {
            foreach $key3 (sort keys %{ $Counts{$key1}{$key2} } ) {
                print $key1,"\t",$key2,"\t",$key3,"\t",$Counts{$key1}{$key2}{$key3},"\n";
            }
        }
    }
}

#########################################################

close ORIGITDB;

exit;
```